UTILIZING COMPUTATIONAL TECHNIQUES TO DESIGN A HYBRID VEHICLE FOR THE ECOCAR MOBILITY CHALLENGE

A Thesis
Presented to
The Academic Faculty

by

Christian Free

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in
Mechanical Engineering

Georgia Institute of Technology
December 2020

UTILIZING COMPUTATIONAL TECHNIQUES TO DESIGN A HYBRID VEHICLE FOR THE ECOCAR MOBILITY CHALLENGE

Approved by:

Dr. Michael Leamy, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. David Taylor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Kenneth Cunefare
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: December 2, 2020

Dedicated to past, present, and future members of the Georgia Tech EcoCAR team.

ACKNOWLEDGEMENTS

I would like to thank the sponsors, mentors, and organizers of the EcoCAR Mobility Challenge for providing an opportunity for students to grow in their fields. I would like to thank Jason Barnes, Dr. Cunefare, and the Georgia Tech EcoCAR faculty advisors for supporting these students in the Student Competition Center. Special thanks go to the members of the EcoCAR team for their hard work and success towards growing as engineers. Lastly, I would like to thank my family for supporting me throughout this process.

This thesis documents work performed by the author in collaboration with several members of the EcoCAR Team. Vehicle level decisions were made by the core leadership including Noah Schaich, Nishan Nekoo, Jason Calvert, Sterling Smith, and Brianna Thornton. The P4 MGU mount was engineered in collaboration with Eamon Flaherty. The Fuel Tank was engineered in collaboration with Nabil Khan, Liza Sheen, and Dang Nguyen.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BSFC	Brake Specific Fuel Consumption
CAV	Connected and Automated Vehicle
CAD	Computer Aided Design
CNC	Computer Numerical Control
DP	Dynamic Programming
ECMS	Equivalent Consumption Minimization Strategy
EM	Electric Machine
EMC	EcoCAR Mobility Challenge
FEA	Finite Element Analysis
FOS	Factor of Safety
GM	General Motors
GT	Georgia Tech
HAZ	Heat Affected Zone
HDS	Hybrid Design Services
HV	High Voltage
ICE	Internal Combustion Engine
MaaS	Mobility as a Service
MGU	Motor Generator Unit
MPG	Miles Per Gallon
OEM	Original Equipment Manufacturer
RPM	Rotations Per Minute
SOC	State of Charge

VDP Vehicle Development Process

SUMMARY

This thesis documents computational techniques and results used in designing a shared-mobility hybrid electric vehicle developed for the Georgia Tech EcoCAR Team, a collegiate engineering team participating in the EcoCAR Mobility Challenge. The competition challenges 12 university teams, 10 from the United States and 2 from Canada, to hybridize a 2019 Chevrolet Blazer and upfit it to SAE Level 2 autonomous operation, primarily for the Mobility-as-a-Service market. The formation and use of dynamic programming for selecting a hybrid architecture is first detailed. The architecture chosen for the competition is then introduced and a selection of custom components engineered for the vehicle is documented. These include a P4 motor mount using CNC machining and topology optimized weldments, a custom 6061-T6 aluminum fuel tank with topology optimized tabs and multiple revisions, and a high voltage A/C compressor mount made with topology optimized weldments and rubber bushings. These efforts help the Georgia Tech team to quickly make improved design decisions that increase vehicle fuel economy.

INTRODUCTION

1.1 The EcoCAR Mobility Challenge Competition at a Glance

The EcoCAR Mobility Challenge (EMC) is the current of several Advanced Vehicle Technical Competitions that have tasked students with engineering cutting edge, prototype vehicles to encourage their growth as engineers. Student teams from 12 North American universities are tasked with transforming a donated 2019 Chevrolet Blazer into a hybrid, semi-autonomous vehicle for the Mobility-as-a-Service (MaaS) market over four years. The vehicle comes with an industry standard internal combustion powertrain which must be replaced with a new hybrid system, complete with all the controls that enable this. The MaaS market constitutes a shift in the automotive industry from personally owned vehicles that are driven and cared for by individuals to fleet owned vehicles operated by a company and rented to consumers. The first year of the competition's vehicle development drocess (VDP) involves selecting the systems and architectures that will provide the hybrid and Connected and Automated Vehicle (CAV) framework for the vehicle. The second year targets the vehicle running with the student designed hybrid systems. The third and fourth years involve the refinement of these systems to increase fuel efficiency and further integrate CAVs technology to build on the autonomy of the vehicle. Goals for autonomy include SAE Level 2 Automation, which involves autonomous steering, acceleration, and braking albeit with regular involvement from the driver. The research presented in this thesis is a part of the competition's 1st and 2nd years. Sponsors of this competition include the U.S. Department of Energy, MathWorks, General Motors, and a variety of other automotive interested companies.

The Georgia Tech (GT) EcoCAR Mobility Challenge team is composed of undergraduate and graduate students primarily studying mechanical engineering, electrical engineering, and computer science. The team numbers approximately 40 students depending on the semester with 3 faculty advisors and 1 General Motors (GM) employed mentor. Participants get research or class credit for their involvement, which varies from engineering and installing components on the car to writing the algorithms that combine data from radars and cameras to enable autonomy. The team's goal is to succeed at the competition's task of building a hybrid vehicle for the MaaS market to the best of the team's limited resources. This goal forms the framework for the engineering challenges and vehicle integration experience that enriches the students' education. The team's vehicle, the GT Blazer, is a P0P4 hybrid with a propulsion and control system designed and built by student engineers.

This thesis is written in compliment to Noah Schaich's thesis "Characterization of a Hybrid Electric Mobility as a Service Vehicle." His thesis discusses in-depth electric vehicle hybridization, the MaaS market, equivalent consumption minimization strategy (ECMS), and some components designed for the Georgia Tech EcoCAR Blazer. This thesis does not include as much detail on the above topics. However, some hybrid vehicle terminology will be discussed for the purposes of presenting the dynamic programming model and other components used in the Georgia Tech EcoCAR Blazer.

1.2 Hybrid Vehicle Terminology

A hybrid vehicle is a vehicle that utilizes two different forms of propulsion, be they a jet propulsion with an auxiliary flywheel or the standard internal combustion engine

(ICE) and electric machine (EM) combination typically found on modern cars. Hybridization in the modern automotive industry is driven by an increased focus on improving emissions and fuel economy. The ICE, EM, and batteries on modern cars can vary in size and location along the powertrain. Plug-in hybrids can plug into an electricity outlet and store energy in big batteries. This usually enables them to operate in an electric only mode without the vehicle producing emissions or burning fuel. The GT Blazer uses a smaller battery and is not designed to be plugged in; thus, the ICE is usually running. Parallel hybrids, as opposed to series hybrids, use any combination of the ICE and EM propulsion to move the vehicle, so long as both systems have a mechanical path to the wheels. The GT Blazer is a parallel hybrid, with a motor and engine in the engine bay powering the front wheels and an additional, bigger motor on the rear axle powering the rear wheels.

Parallel hybrids can place their EMs in a variety of positions. The motor position is typically referred to with a P0, P1, P2, P3, or P4 nomenclature. This nomenclature is drawn in Figure 1 where P0 attaches to the ICE belt drive, P1 and P2 attach on either side of the clutch uniting the ICE and transmission, P3 attaches to the driveshaft powering the same wheels as the ICE, and P4 powers a different axle than the ICE. Each location has advantages and disadvantages. In short, a P0 hybrid requires minimal modification to the industry standard vehicle and offers small improvements in fuel economy and function. P1 and P2 motors can be packaged easily and offer moderate performance but require packaging considerations with the engine and transmission. P3 motors require space along the driveline to splice in and do not get the benefit of gearing through the engine's transmission but can be substantial in size. P4 hybrids can be simply integrated onto a

separate axle with good performance but cause additional tire wear when absorbing energy from the engine. The GT Blazer is a P0P4 hybrid by this nomenclature.

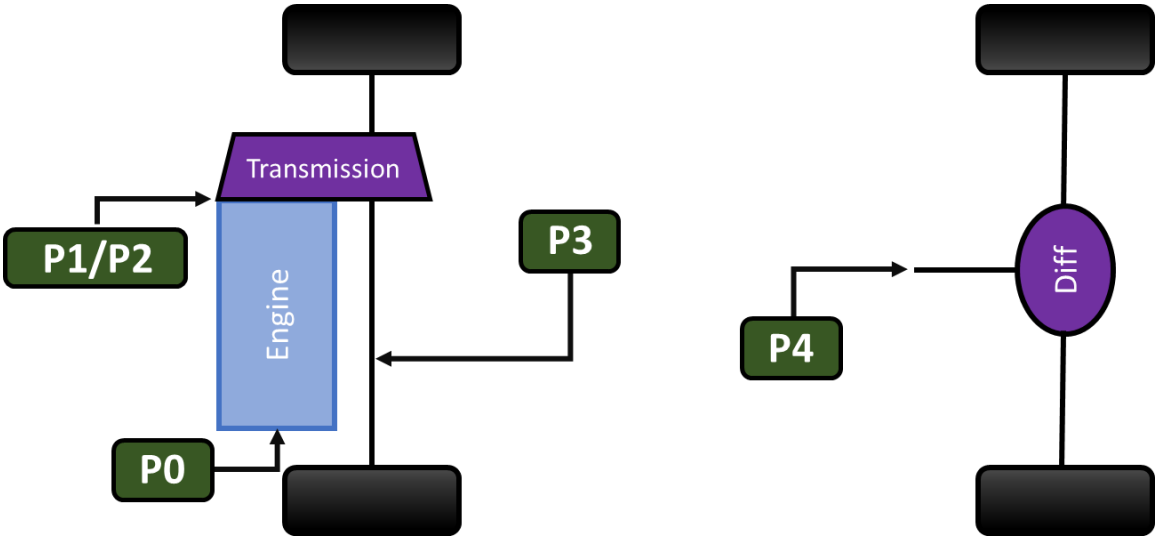

Figure 1: Electric motor P0, P1, P2, P3, P4 placement map

Attaching an EM to an ICE allows the car to be more fuel efficient through a variety of operating modes, perhaps most notably through "load point shifting." To understand load point shifting, it is crucial to understand that a car's ICE has varying fuel efficiency depending on what rotations per minute (RPM) and power output (usually determined by accelerator pedal position) the engine is operating at. A contour graph of brake specific fuel consumption (BSFC) vs engine power and RPM is shown in Figure 2 as an example of an engine's BSFC. The contour comes from *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles*, which goes in depth into the technical details of hybrid vehicles [1]. BSFC describes the mass of fuel it takes to generate a unit of power. Driving down any given road requires a certain amount of power to prevent slowing down due to aerodynamic and frictional loads. The most efficient way an ICE can deliver that power is by changing the rpm and load to minimize BSFC. On a typical vehicle, the transmission will shift into the

lowest gear that can deliver the needed power. For example using Figure 2, if 40 kW is needed it would require 320 g/kWh and 500g /kWh to operate at 3000 and 4000 RPM, respectively. The engine would burn less fuel if it operated at the lower RPM. On a hybrid vehicle, an EM can absorb excess power from the engine and store it in the battery for later use. Using Figure 2, if 40 kW is needed and the engine is operating at 3000 RPM, the ICE could deliver 70 kW with the electric motor absorbing 30 kW. The ICE would spend a bargain 255 g/kWh of fuel, storing the excess energy in the battery. Later, the ICE could be potentially turned off and the EM could use this energy to propel the car.

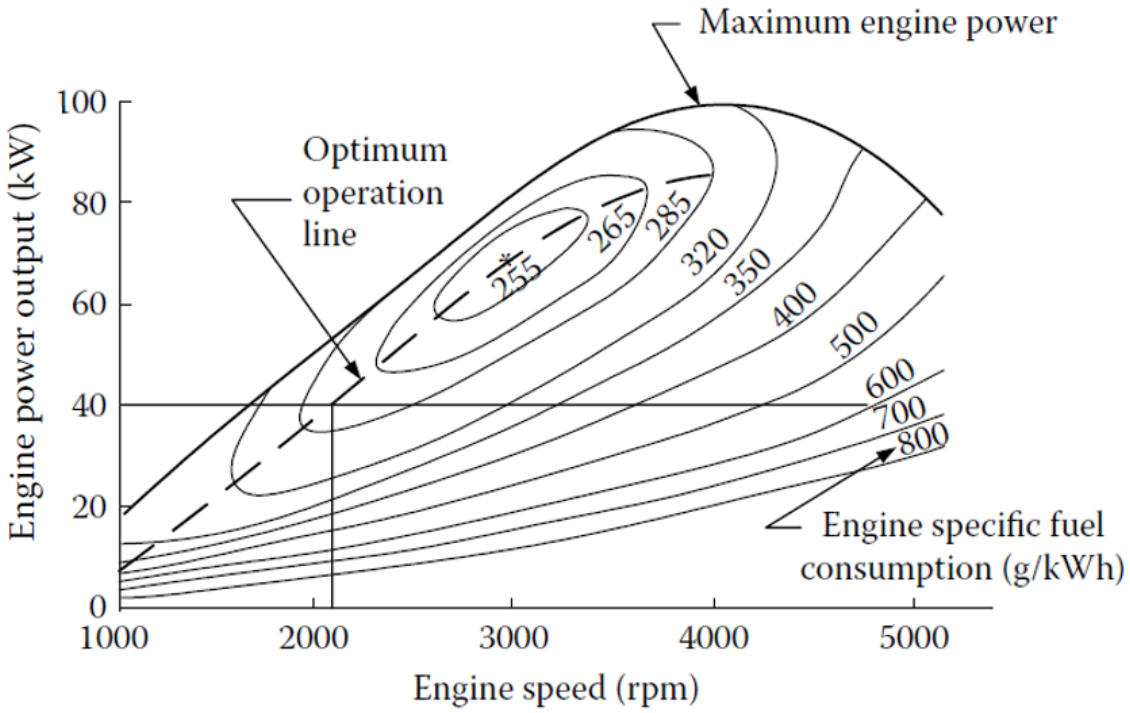

Figure 2: Contour graph of engine BSFC [1]

Load point shifting is one of many operating modes a parallel hybrid can perform. Other ways a hybrid system can be used include hybrid traction, where both the ICE and EM supply tractive power; ICE only traction; EM only traction; regen braking, where the

EM is dragged to charge the battery; and engine charging, where the car is not propelled and power from the ICE is absorbed by the EM and transferred to the battery.

Determining how to direct the two propulsive systems in a hybrid vehicle is a significant controls problem. If driving on a road at a fixed speed, the vehicle could use load point shifting until the battery is charged, then use EM only traction until the battery is discharged. If the vehicle is approaching a hill, it might be most efficient to discharge the battery with EM only traction. This would make room in the battery for energy that could be absorbed while going down the hill. Complicating this problem is the fact that consumer vehicles rarely stay at the same speed and are constantly accelerating and decelerating to meet traffic flow. A drive cycle is usually created to test control strategies to these dynamics. For the proceeding models, the drive cycle is a series of vehicle velocity versus time that represents typical velocities a vehicle might see as it commutes to a city.

For the GT Blazer, two controllers are in development to direct the vehicle power in an efficient manner. The first is a rules based controller, which uses simple rules such as if the battery state of charge (SOC) is high then use EM traction only and if SOC is low then use regen braking and load point shifting. This controller would be easy to implement but might not give optimal fuel efficiency. The second is ECMS and is an algorithm that determines if it would be better to supply the requested power with the ICE or EM, based on the fuel that would inevitably be consumed to provide the power [2]; in a non plug in hybrid, all the energy in the battery derives from either driving down hills or absorbing power generated by the ICE. As presented in [3], the driving equation of ECMS is

$$H = P_{fuel} + s * P_{battery} * p(soc) + p(feasbility) \quad (1)$$

where the control strategy minimizes the weighted cost, *H*, it would take to provide propulsive power. In the equation, P_{fuel} is the ICE power, s is fuel equivalence factor and is determined by the engineer or algorithm, $P_{battery}$ is the battery power, $p(soc)$ is a cost multiplier as a function of SOC, and $p(feasbility)$ is a cost penalty associated with making undesirable control choices, such as shifting gears too much or over extending the battery. These values must either be optimized for the powertrain architecture and drive cycle or adapt in real time to produce the best possible fuel economy. The additional control complexity of ECMS usually provides better fuel economy than rules based control.

ARCHITECTURE SELECTION AND DYNAMIC PROGRAMING

2.1 Initial Architecture Options

The GT Blazer's initial architecture options were determined by a review of the MaaS market's customer requirements, EcoCAR Mobility Challenge battery and engine options, and packaging space in the 2019 Chevrolet Blazer. Ride hailing drivers were interviewed as prospective customers of the car and they desire good fuel economy [3]. In support of the EcoCAR Mobility Challenge, engine/transmission options were offered in a non-modifiable, GM supported "power cube." Providing teams fully functioning power cube options would allow them to finish the vehicle faster and move onto CAVs development and success in the competition. Other engine and transmission options were allowed by the competition, but GM would not provide engineering support during their integration. Similarly, battery pack options were restricted by the competition to black box solutions by reputable companies, effectively narrowing battery options to the GM offered and supported HEV4 pack and a custom pack engineered by Hybrid Design Services (HDS). This restriction was done to increase high voltage (HV) safety in the competition and accelerate vehicle development towards CAVs refinement. Sponsor donated powertrain components were identified as being most likely to facilitate the team's success due to available technical support and a general reduction in the expertise required for component integration. This narrowed power cube and HV battery options to those supplied by GM. The GM sponsored engines that the team pursued were the 1.5 liter turbocharged LYX engine, 2.0 liter turbocharged LTG engine, and 2.5 liter LCV engine. All three engines have four cylinders and run on gasoline. The offered power cubes all

mounted transverse in the Blazer, making packaging of a P3 EM on the front half shafts difficult. P1 and P2 EMs would require modification and splitting of the GM power cubes. However, there was still room to fit a P0 EM on the belt drive of the engine and a P4 EM on the rear axle. This narrowed the team's architecture selection to P0 and P4 EMs that could utilize the power provided by the HEV4 battery pack.

2.2 Modeling and Dynamic Programing Overview

A Simulink model with an adaptive ECMS controller was created to predict the performance of prospective architectures. However, ECMS is not ideal for architecture selection because the cost parameters need to be optimized for each architecture before their peak fuel economies are obtained. In parallel, a dynamic programming model of a hybrid electric vehicle was created to better compare architectures with respect to fuel economy. The code for this model is included in the Appendix. Confidential specifications for the 2019 Blazer have been omitted.

The team developed a dynamic programming (DP) model to fairly rank the efficiencies of architectures with the guarantee that an architecture is not poorly represented by a subpar control strategy. The disadvantages of DP are increased run times and lower model fidelity due to its backwards-looking nature. The DP script is written in MATLAB and is ran by inputting a drive cycle velocity trace and vehicle parameters, such as engine torque, component masses, battery state of charge range, and motor generator unit (MGU) efficiency. It is a backwards-looking simulation that sweeps through every combination of vehicle states – namely engine gear, engine throttle, and MGU torque – and determines the path of control decisions that minimizes fuel usage.

The framework used to make the model followed Pei and Leamy's paper, "Dynamic Programming-Informed Equivalent Cost Minimization Control Strategies for Hybrid-Electric Vehicles" [4]. In this paper, DP cost *J* is minimized over a time model with *k* being the time index. A state variable x_k at each time step is changed by one or more control variables u_k. Penalties ϕ_k are added to cost if a control method is undesirable. The total cost, *J*, of the optimization problem is found using

$$J_{0,\pi}(x_0) = g_0(x_0) + g_N(x_N) + \phi_N(x_N) + \sum_{k=0}^{n-1} [h_k(x_k, u_k) + \phi_k(x_k, u_k)] \qquad (2)$$

where π is a particular control strategy, $g_{0/N}(x_{0/N})$ is the initial and final cost at the initial and final state, $\phi_k(x_k)$ is the penalty at the time *k*, and $h_k(x_k, u_k)$ is the cost function with respect to the state and control variables.

Pei and Leamy iterate through each state during each time step. At each time *k*,

$$J_k(x_k^j) = \min [J_{k+1}(x_{k+1}^j) + h_k(x_k^j, u_k) + \phi_k(x_k^j, u_k)] \qquad (3)$$

is used to calculate to calculate cost at a state *j*. The forward step cost, J_{k+1} is added to the current cost and penalty function. The minimum of all possible states *j* is taken to be the final cost at $J_k(x_k^j)$. After iterating through the entire state-time matrix, a minimal cost will be found at the initial state. This works because of Bellman's Optimality Principle, where the "optimality of the past action has no effect on the optimality of the future action" [4]. In this way, if each state knows the optimal cost forward, working backwards from the final to initial time will produce a global optimum.

In the GT EcoCAR DP model the overarching equations for required force, *F*, and power, *P*, at each time step in a drive cycle are

$$F = A + Bv + Cv^2 + (\delta m)ma \quad (4)$$

$$P = Fv + l \quad (5)$$

where *A*, *B*, and *C* are confidential coast down coefficients for the Blazer, *v* and *a* are velocity and acceleration at each time step, respectively, δm is the mass factor, *m* is the mass of the vehicle, and *l* is accessory load. All model decisions stem from the requirement for the vehicle to meet the power demand for a given time step on the drive cycle.

The cost, *J*, in the GT DP model is grams of fuel. The state variable, *x*, is the battery SOC. The time step *k* progresses at 1 second intervals, which is the same resolution as the EMC drive cycles. The control variables, *u*, meeting the required power are engine torque, transmission gear, and optionally α, a variable for splitting the torque between multiple motors. Utilizing fewer variables significantly improves computation time, so the number of variables was minimized. For a single motor parallel hybrid, only engine torque and transmission gear are needed as control variables. It is assumed that the MGU fills in any power missing or absorbs excess power from the engine via the battery power, following:

$$P_{battery} = P_{required} - P_{engine} \quad (6)$$

If two motors are utilized in the parallel architecture, α splits $P_{battery}$ by

$$P_{MGU1} = P_{battery} * (1 - \alpha) \quad (7)$$

$$P_{MGU2} = P_{battery} * (\alpha) \quad (8)$$

$$0 \leq \alpha \leq 1 \quad (9)$$

A limitation of the GT DP model is that α is only varied from 0 to 1, which means both motors are either both generating or both absorbing. In effect, this biases power to the more efficient MGU. With two MGUs on a vehicle, it would be possible for one MGU to

generate while the other motors. This was not explored and could provide further improvements in efficiency.

A visual representation of the final steps in a dynamic programming model is shown in Figure 3. The model would progress backwards from time = N and the Final SOC state. It would calculate the cost it takes to get from each SOC in N-1 to the final SOC and store the information for each SOC in N-1. For example, it costs 1 and 4 units to get from 0.6 and 0.4 to the final SOC, respectively. Next it would progress back a time step. Here it would sum the cost to move from N-2 to each SOC in N-1 with the stored cost for each SOC in N-1. This is only shown at time = N-2 for SOC = 0.5. Finally, it would save the minimum cost it found, and this would represent the minimum cost to move from the SOC in N-2 to the final SOC. If the vehicle started at SOC = 0.5 and N-2, the optimal progression would be to move from 0.5 SOC at N-2, to 0.5 SOC at N-1, to the final SOC. This would result in a global cost of 5. It can be visually seen that this is the minimum cost for this starting SOC.

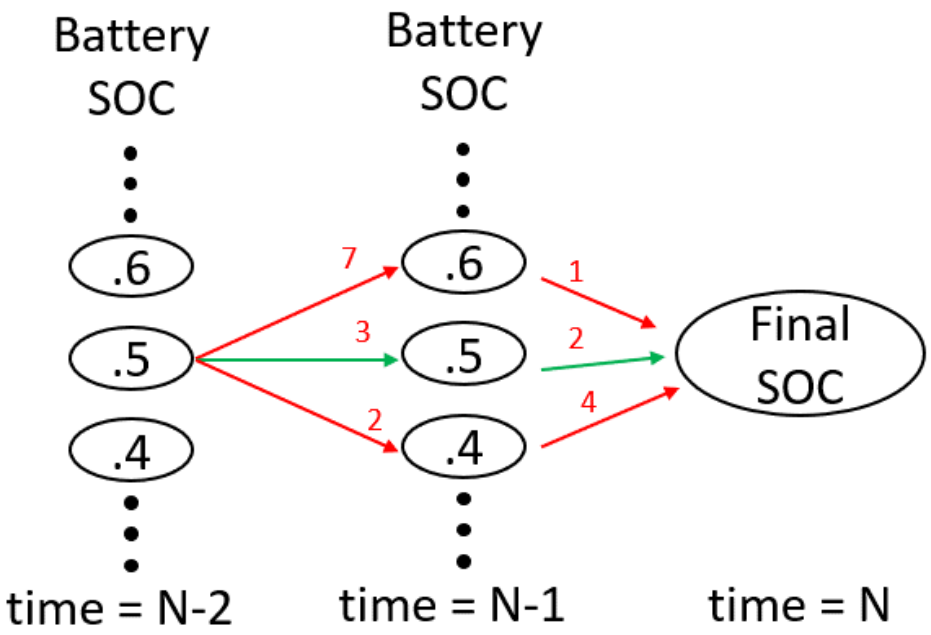

Figure 3: Depiction of a dynamic programming path progression illustrating the cost to travel between SOC points

An example of a DP run drive cycle is shown in Figure 4, which shows the propulsion system control strategy for a P0P4 hybrid over the EMC City drive cycle. Battery SOC, shown in the bottom subplot, is optimized for fuel consumption over the drive cycle. The control decisions ICE throttle, transmission gear, P0 throttle, and P4 throttle are shown in the top and middle subplots.

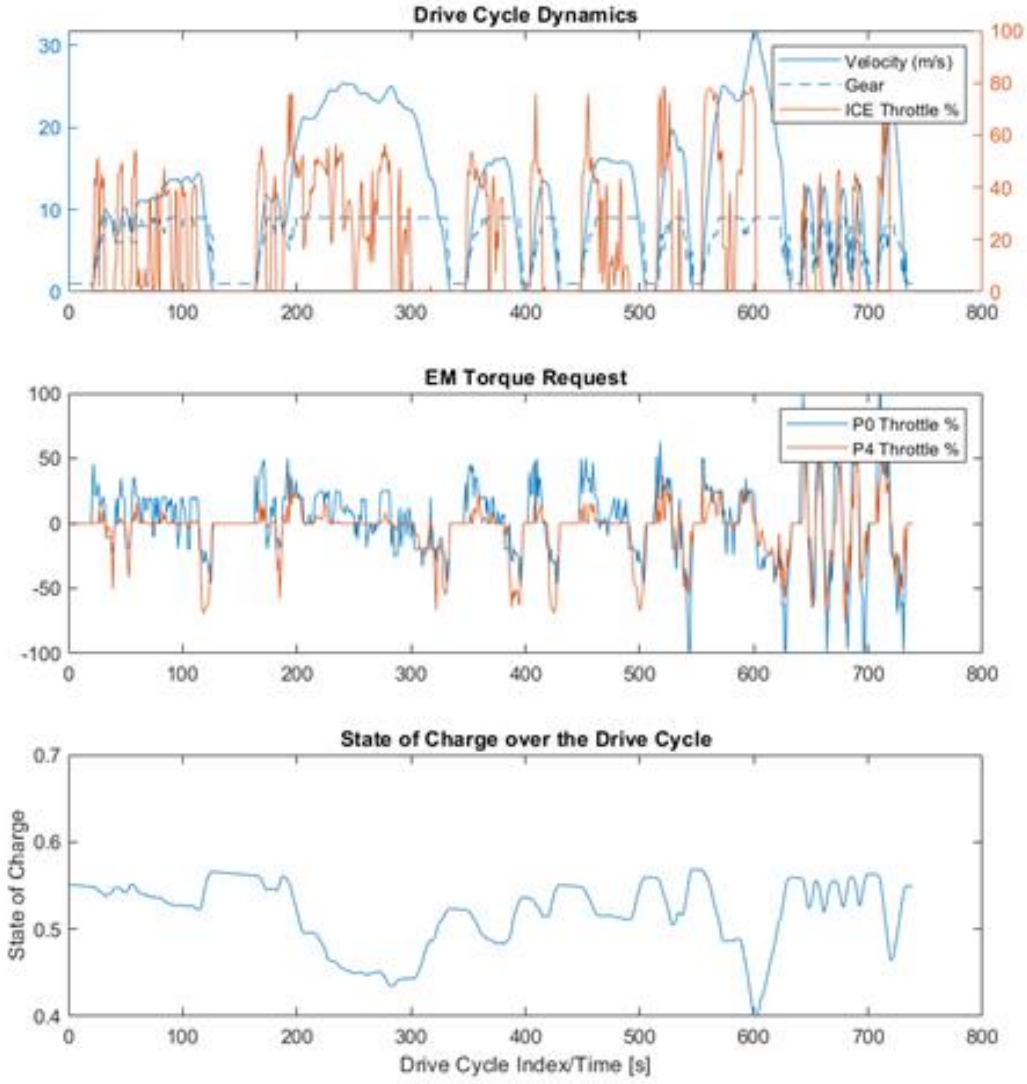

Figure 4: Powercube controls (top), EM torque request (middle), and battery SOC (bottom) dynamic programing controls for a P0P4 hybrid on the EcoCAR city drive cycle.

2.3 GT Dynamic Programming Model In-Depth

2.3.1 Model Components

The model uses several GM confidential arrays and metrics whose figures will not be disclosed. However, the input information will be described, and non-confidential results will be shown. Input information includes masses added and removed for key powertrain components such as engines, transmissions, and EMs. An accessory load is added that was estimated from experimentally measured accessory loads. Transmission efficiency as a function of speed and torque is input along with gear ratios for the transmission. Engine torque and fuel are input as a function of RPM and throttle. The EMs are assumed to have a maximum torque and a maximum power, simplifying their torque curve into these two parameters. EM efficiency as a function of speed and torque is also input but is assumed to be the same in motoring and generating. The battery is modeled as having a maximum power it can deliver and absorb as well as constant voltage, resistance, and capacity. An SOC range is given, outside of which control decisions are penalized. Similarly, a start and end SOC is given to ensure the vehicle operates in a charge sustaining mode.

These choices in input data result in a simple model that has the accuracy to make architecture decisions based on component efficiency. Key assumptions and their defenses are outlined next. The MGU max torque curve is modeled with low fidelity; it consists of a constant torque region and a constant power region. Figure 4 above shows that MGUs are usually only at peak output during low speeds where energy use is low, so this simplification has minimal effect on total energy consumption. The battery is modeled

using constant voltage and resistance. Given that the GT team anticipates using an SOC range between 40% and 70%, the real voltage range will only be 10V on a 300V battery and this simplification is reasonable. Inertia is ignored, apart from the mass factor, and no transient effects are considered. This was deemed acceptable for comparing architectures because it was assumed transient effects would be similar between architectures. However, this may not be true for turbocharged engines with significant transient power changes. Likewise, regen braking was assumed to max out the MGU negative torque capacity before the mechanical brakes are used. This produces optimistic fuel economy estimates that should be similar across architectures. Several optimistic assumptions are made in the transmission. There is no torque converter slip, no neutral gear, and gear shifts are instant. Thus, whenever the transmission is not spinning the engine is not spinning. A result of this transmission model is that P0 MGUs can perform unnaturally efficient regenerative braking by generating at low engine speed when there would normally be torque converter slip. This gives P0 architectures an unrealistic increase in efficiency. Utilizing a P0P4 architecture minimizes the impact of this simplification on real fuel economy because the P4 MGU will be able to brake the vehicle regardless of engine speed.

An important aspect of the dynamic programming results presented in this paper is that the MPG estimates are in general optimistic and unfeasible. The model is omniscient to the entire drive cycle and thus can prepare for future power requirements that a real vehicle would not be able to know – unless the drive cycle was planned. It also can make decisions that would be unacceptable for vehicle ride and handling, such as alternating between gears quickly.

2.3.2 Model Convergence Study

A convergence study was run to determine appropriate state and control variable sizes. A table illustrating this study for P0 and P4 configurations is shown below in Table 1. A state size of 5000 and a control size of 100 provides acceptable resolution for both the P0 and P4 architectures. A separate convergence study was done for P0P4 architectures and the results are shown in Table 2. It is shown that an increase in α size increases solve time with no benefit to efficiency accuracy. The state and control variable sizes used for the remainder of the report are 5000 for the SOC, 100 for engine throttle, 9 for transmission gears (all transmissions had 9 forward gears), and 20 for α.

Table 1: Dynamic programming convergence study for P0 and P4 architectures

	State x Control				
	500 x 20	**5000 x 100**	**10000 x 500**	**15000 x 500**	**15000 x 750**
P0 [mpg]	44.0	38.4	38.2	38.1	38.1
P4 [mpg]	45.0	39.5	39.3	39.2	39.2

Table 2: Dynamic programming convergence study for P0P4 architecture

	State x Control x α				
	500 x 20 x 20	**500 x 20 x 50**	**500 x 20 x 100**	**500 x 50 x 20**	**500 x 50 x 50**
P0P4 [mpg]	45.2	45.2	45.2	46.2	46.2
Solve time [s]	64	126	227	124	268
Time increase [vs 500 x 20 x 20]		2.0	3.5	1.9	4.2

2.4 Architecture Selection using Dynamic Programming Model

Two-variable parametric sweeps were used to explore the impact of various component choices on a P4 architecture's fuel economy. Sweeping two variables at a

time produces easy to read two dimensional contour plots of complimentary parameters, such as EM power and battery capacity. A simple P4 architecture was chosen due to its lack of computational complexity. A low fidelity model was used because the study only needed to be accurate for ranking. Ranges for these sweeps were set to be within the feasibility range of the GT team. For example, vehicle mass was swept with a range to 200 kg on either side of the GT team's anticipated weight. A 200 kg reduction in weight would require a large effort by the team, but it might be worth it if such a reduction increases fuel economy substantially. Step size of the sweeps were iteratively refined on until the curvature of the contour was illustrated. Sweeps were performed to obtain fuel economy for battery capacity versus EM power, vehicle mass versus battery capacity, EM torque versus EM power, EM gear ratio versus EM power, vehicle mass versus vehicle drag coefficient, vehicle mass versus EM power, and vehicle mass versus vehicle rolling resistance. These sweeps show the interdependence and impact of these variables with respect to fuel economy. Due to the optimality of dynamic programming, each of these studies make optimal control decisions for their architecture parameters. A selection of these sweeps are shown in **Figure 5** and **Figure 6**, where contours represent fuel economy. The red dots show where a GT P4 architecture with the HEV4 battery and a 50 kW P4 MGU would be. Insights from these graphs include the importance of having a balance between MGU gear ratio and power output. Large engineering efforts aimed at integrating big, powerful MGUs do not have a good value proposition due to their steep drop off in returns after 35 kW. Decreases in vehicle mass would be helpful for fuel economy but would not yield large improvements within the mass range the GT team could remove from the Blazer. Increasing battery capacity was also considered. Although

doubling the battery capacity provides a significant 3 mpg improvement, increasing the battery size would inevitably increase vehicle mass which would partially counteract this potential increase in efficiency.

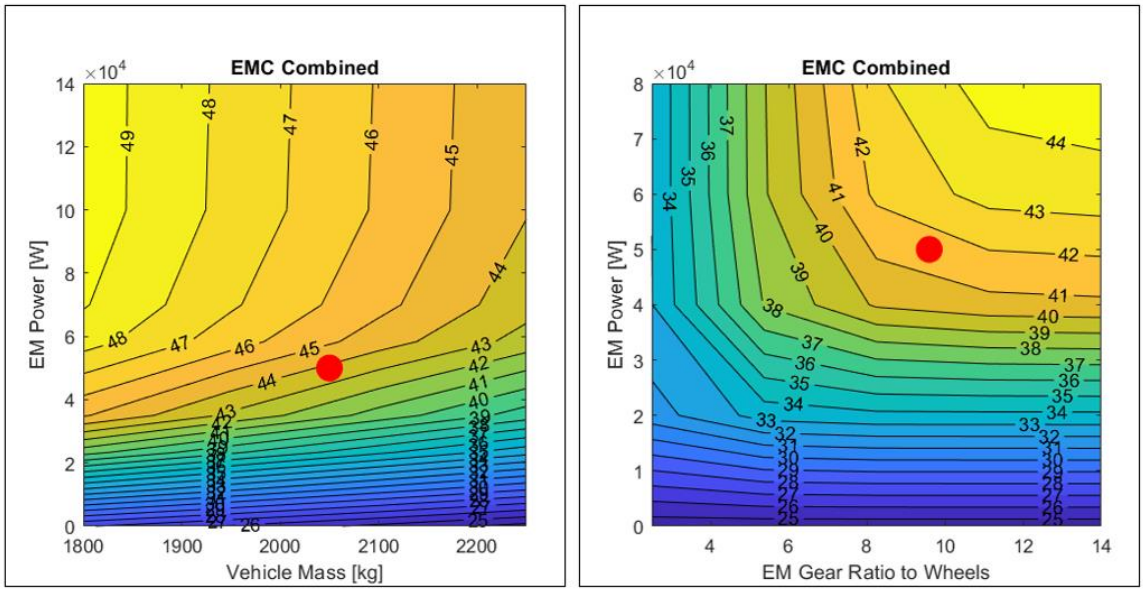

Figure 5: Dynamic programing fuel economy contours showing EM power vs vehicle mass and EM power vs gear ratio for a P4 hybrid

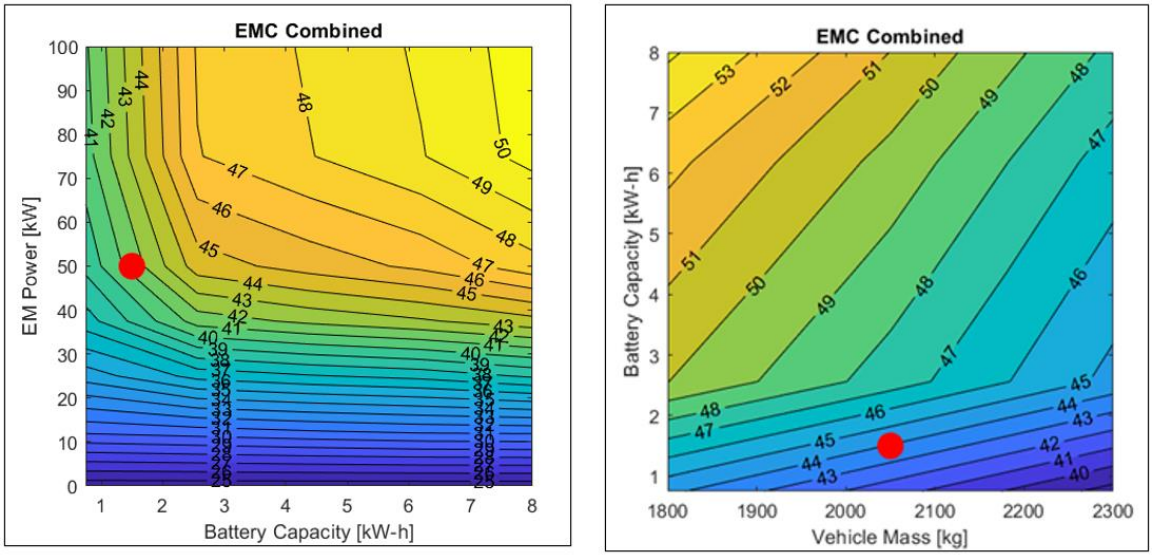

Figure 6: Dynamic programing fuel economy contours showing EM power vs battery capacity and battery capacity vs vehicle mass for a P4 hybrid

The team's low risk appetite steered the team towards the GM provided battery pack and 50 kW P4 MGUs. From these sweeps, the team also identified that having a gear ratio around 10 is a simple way to improve fuel economy, as opposed to direct drive. This directed design towards P4 MGUs with integrated transmissions. Additionally, sweeps including rolling resistance and air resistance as parameters showed that putting effort into choosing low rolling resistance tires or improving underbody airflow could be relatively low effort tasks that improve fuel economy by significant amounts. These sweeps are not shown to protect the confidentiality of rolling resistance and aerodynamic drag for the vehicle. An additional conclusion influencing architecture selection was that large engineering efforts aimed at integrating powerful MGUs do not have a good value proposition. Increasing EM power from the expected value of 50 kW would only increase fuel economy by a maximum of 2 mpg. The larger motors needed for larger EM power would require substantial modification and reinforcement to the trunk and subframe to enable their packaging.

At this point in the architecture selection process, the team had narrowed down P4 MGU components to the Bosch SMG 180/120, the AAM EDU2, and the Magna eRAD. Each of these MGUs are produced by competition sponsors, fit reasonably well in the rear of the Blazer, and have integrated transmissions. The Denso ISG, a sponsored MGU, was also considered as a P0 motor, either by itself or in a P0P4 architecture. Among these options, engine selection was found to have the biggest impact on vehicle fuel economy and performance. DP results in Figure 7 showed that, for similar architectures, the LYX engine achieved as much as 4 mpg better than the LTG engine with the LCV engine

producing close results to the LYX. Each red band is an engine, with P0 on the right, P4 architectures in the middle, and P0P4 architectures on the left.

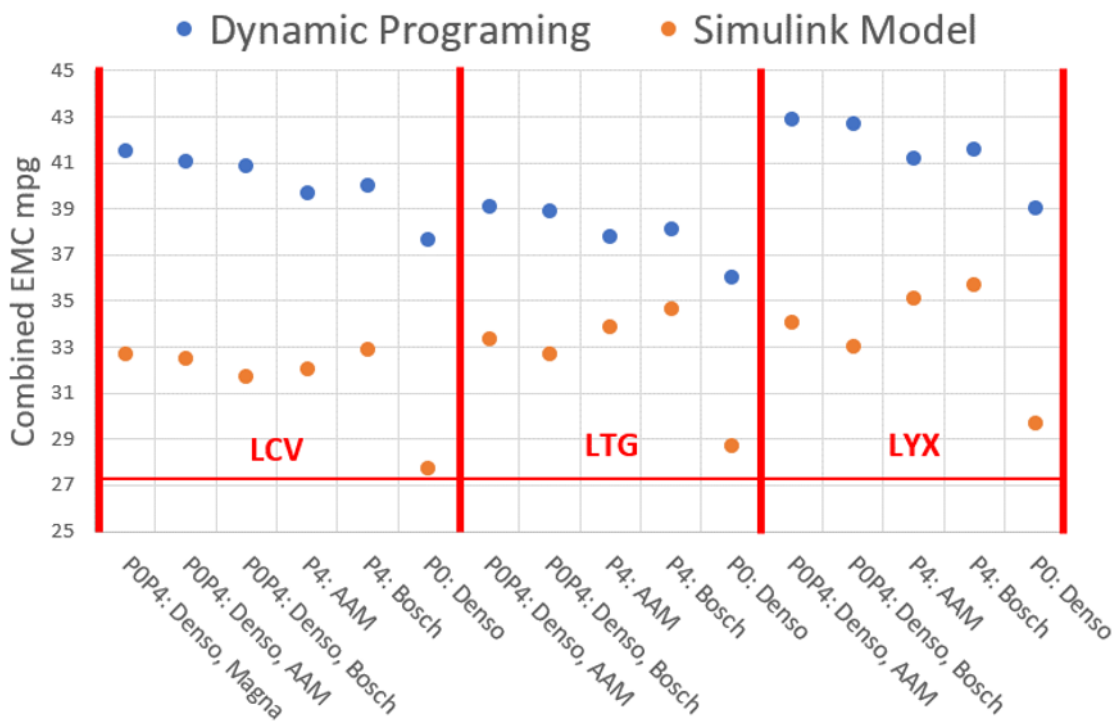

Figure 7: Simulink ECMS model vs. dynamic programing fuel economy results for selected architectures

Figure 7 also gives a comparison between the dynamic programing and Simulink ECMS model results for various architectures. For DP, the P0P4 LYX engine performed the best, with the LCV P0P4 close behind. The P0 performed the worst for both Simulink and DP, which makes sense because the motor has comparatively low power output. DP performs much better than Simulink, which is expected because it is an omniscient controller. However, the trends do not line up exactly between Simulink and DP. For Simulink, the P0P4 architectures do worse than their P4 counterparts, which is not logical. A P0P4 should perform slightly better than a P4 because it has opportunities to shift the

load points between the two MGUs. If it is assumed that there is a controller error in the Simulink model and the P0P4 architectures in fact do better than their P4 counterparts, then the trends for each architecture within each engine line up well between DP and Simulink.

2.5 Architecture Selection

The fuel economy results gleaned from dynamic programing fed into the fuel economy and cost to consumer sections of an evaluation matrix. A snippet of the evaluation matrix is shown in Table 3 with only the LCV engine options shown. Also considered in the evaluation matrix were the LTG engine, the LYX engine, and the HDS battery pack. Points were awarded on a scale of one to five with five being better than one. Fuel economy was calculated and then normalized to this scale. EcoCAR Cost to Consumer and Cost to Team were calculated and then inversely normalized to this scale. Of the metrics considered in the evaluation matrix, Fuel Economy is weighted the highest at 25% due to the team's desire for competition success and educational value. Fuel economy is expected to comprise a large portion of competition points. Furthermore, focusing on high fuel economy technologies increases the relevance of the students' experience to the automotive industry. Fuel economy is evaluated by increase in fuel mpg over the stock LGX Blazer and is calculated using Dynamic Programming over the EcoCAR drive cycles. Risk is weighted second highest at 20% due to the team's desire to finish the vehicle early and have reliable operation at competition. Risk was quantified based on the number of added components and failure points. Architectures with turbochargers, custom battery packs, and custom transmissions were perceived to add failure points and received worse risk scores. Finishing the propulsion system early guarantees the students see the entirety of the VDP. Educational Value is given a higher weighting than Engineering Complexity. This

penalizes highly complex architectures but allows for architectures with a high value-to-complexity ratio to be better represented.

Table 3: Snippet from evaluation matrix used to select architecture

		Magna eRAD P4, Denso ISG P0 LCV	AAM EDU2 P4, Denso ISG P0 LCV	Magna eRAD P4 LCV	Denso ISG P0 LCV
Evaluation Criteria	Weight	5 is best, 1 is worst			
Risk - # components, failure points (1-5)	20%	3	2	4	5
Δ Fuel economy (OEM)	25%	4.63	4.25	4.40	3.07
Engineering Complexity (PSI) - Engineering man hours (1-5)	8%	2	2	5	4
Engineering Complexity (CSMS) - Engineering man hours (1-5)	8%	3	2	4	5
Educational Value (1-5)	12%	5	4	1	2
EcoCAR Cost to Consumer	15%	5.00	4.86	4.96	4.13
Consumer Appeal - Performance, etc. (1-5)	8%	4	4	3	2
Cost to Team - Inverters, MGU, HV Battery	4%	4.11	3.22	5.00	4.11
	Average score:	3.99	3.44	3.92	3.67
	Normalized score:	1.00	0.86	0.98	0.92

The architecture determined by the evaluation matrix to be the best option is the Denso ISG P0, Magna eRAD P4, and GM LCV powertrain with the GM HEV4 battery pack. While the decision matrix is a useful tool for comparing a multitude of architectures, this tool primarily served as a way for the team to ensure that the architectures in consideration reflected the values and goals of the team. Most of the LCV and LYX architectures received similar valuations, likely beyond the fidelity of the evaluation matrix. The team's low risk appetite derived the selection of the non turbo-charged LCV engine with the easy-to-integrate Magna eRAD P4. The high value/risk ratio of adding the Denso ISG P0 drove its addition to the propulsion system. The small increases in fuel economy coupled with the target market centric features, such as stationary charging and flying starts, outweigh the added complexity of this architecture.

DESIGN OF VEHICLE COMPONENTS

3.1 Architecture Overview

A pictorial representation of the GT Blazer architecture is shown in Figure 8. The vehicle is a P0P4 hybrid with the Denso ISG EM on the belt drive of the LCV engine. The M3D 9 speed transmission is paired to this engine. A team fabricated fuel tank is mounted under the car in a similar space to the larger, original equipment manufacturer (OEM) fuel tank. The design of the fuel tank will be discussed further in Section 3.4. The Magna eRAD motor is mounted on the rear axle. Above it, in the trunk, is the HEV4 battery pack. The battery pack supplies power to both EMs. Not shown in the picture are the Magna Dual Inverter and the Rinehart PM100DX Inverter, which meter the P4 and P0 power supplies, respectively.

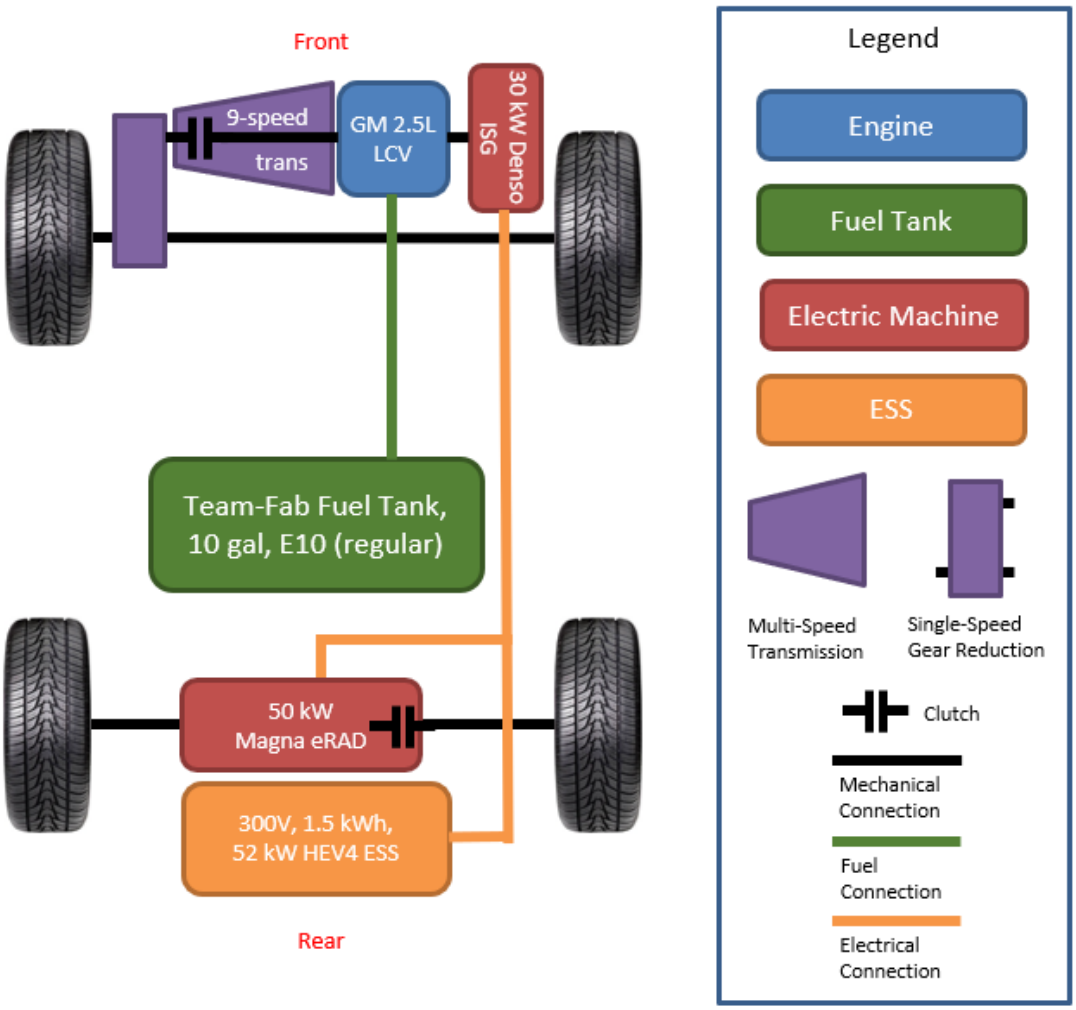

Figure 8: GT Blazer P0P4 architecture overview

Implementing this architecture on the 2019 Chevrolet Blazer chassis requires some modifications. The Blazer arrived at GT with a V6 engine and all-wheel drive. The V6 and driveline were removed. Since a version of the Blazer is offered with the LCV from the factory, the team's LCV engine and transmission mounted directly to the front subframe via OEM mounts. The engine intake uses the OEM LCV intake and requires no modification. The exhaust system uses the LCV downpipe and a modified Blazer LGX

exhaust pipe to have a single exit, smaller muffler. The front half shafts come from the OEM LCV Blazer. The LCV belt line is simplified so that only the P0 motor is attached to the crankshaft pulley. This simplifies P0 integration, which was expected to take significant engineering effort. An electric water pump and HV A/C compressor are needed because of the simplified belt drive system. The engine water pump is mounted to the front subframe. The A/C compressor is mounted to the engine bay frame and will be discussed in Section 3.5.

Attaching the P0 MGU to the LCV engine is done cognizant to some lack of information surrounding the limitations of the LCV engine. GM would not provide exact specification on the crankshaft torsional strength or max crankshaft bearing loads but expressed that the chosen MGU would not create loads more than what the LCV engine could withstand. Furthermore, a belt with appropriate strength for the P0 MGU is yet to be specified. Given the absence of quantitative specification on the LCV accessory drive train and belt selection, more development of the P0 drivetrain is recommended before full torque of the P0 MGU is implemented to protect the components. Until this analysis is complete, the performance of the P0 MGU should be conservatively limited to below what was modeled in DP.

The rear of the car sees more substantial modification. The spare tire in the trunk is removed to make room for the added components, since a spare tire is not needed for the competition. The P4 inverter and multiple electronics are mounted where the spare tire was and the effect of this on crashworthiness is not fully understood. The HEV4 battery pack is mounted just behind the rear seats. The trunk false floor is custom and is raised approximately two inches above the OEM floor to package the HEV4 battery cooling. The

P4 MGU mounts where the OEM rear differential was packaged. A custom mount was designed to facilitate the P4 motor and will be discussed in Section 3.3. Custom rear half shafts were outsourced to Raven Engineering.

In general, efforts were made by the team to design and manufacture as many components as possible in house. This was done to improve students' knowledge of the design for manufacture process. It also facilitated fast iteration of components and lower costs. The manufacturing resources immediately available to the GT EcoCAR team include computer numerical control (CNC) mills and lathes, manual mills and lathes, a waterjet, a 3D printer, a TIG welder, and a selection of small sheet metal forming machines. These resources heavily influenced the designs the team was able to produce.

3.2 Rules for Modifying the Vehicle

The EcoCAR Mobility Challenge implements several rules for modifying structures to not compromise the vehicle's safety and crashworthiness. Notably, all installed propulsion system components must be able to survive a +/- 20 g static acceleration towards the front of the car (x axis), +/- 20 g static acceleration towards the passenger side of the car (y axis), and +/- 8 g static acceleration towards the top of the car (z axis) without entering the plastic region of deformation. This must be accomplished with a minimum 1.5 factor of safety (FOS). Solving the kinematic equation

$$v_f^2 = v_0^2 + 2a\Delta x \tag{10}$$

for acceleration, a 20 g acceleration is the average acceleration experienced crashing at 30 mph in 1.5 ft. The purpose of this rule is to provide suitable structural strength for vehicles

to make it through standard competition forces and fatigue, as well as not endanger people in the event of a moderate to severe crash.

Furthermore, all portions of the chassis are identified as being either red, yellow, or green. This distribution is confidential and not visually depicted in this report. Red and yellow structures are prohibited from being modified significantly. Green structures may be modified as much as a team desires. Red and yellow structures may be modified with organizer and GM approval. This usually necessitates a formal document and evidence to prove that the modified structure is at least as safe as the OEM structure. Little of the structure is green. Much of the supports in the cabin are yellow. The rest of the chassis is red.

Bolting and fastening to all areas of the vehicle requires attention to fastening rules. All holes must be drilled at least twice the diameter of the hole away from another hole, weld, or edge. Holes must be smaller than 13 mm in diameter. Welds must be performed such that stress risers are minimized and the heat affected zone is small. One intention of these rules is to not compromise the normal driving performance of the highly optimized chassis. Some chassis features are designed to be stiff and strong, whereas others are designed to be soft and deform in a crash.

3.3 P4 Motor Mount

3.3.1 Design

The Magna eRAD MGU was packaged in the same location as the stock Blazer's differential as shown in Figure 9. This is convenient for half shaft placement and also

allows for a rules compliant ground clearance of greater than 7”. The mount requires a strong connection to the chassis to withstand the torque of the motor and required crash load cases. The team desired to mount the motor to the chassis using rubber bushings due to concerns that the MGU might transfer uncomfortable vibrations to the chassis

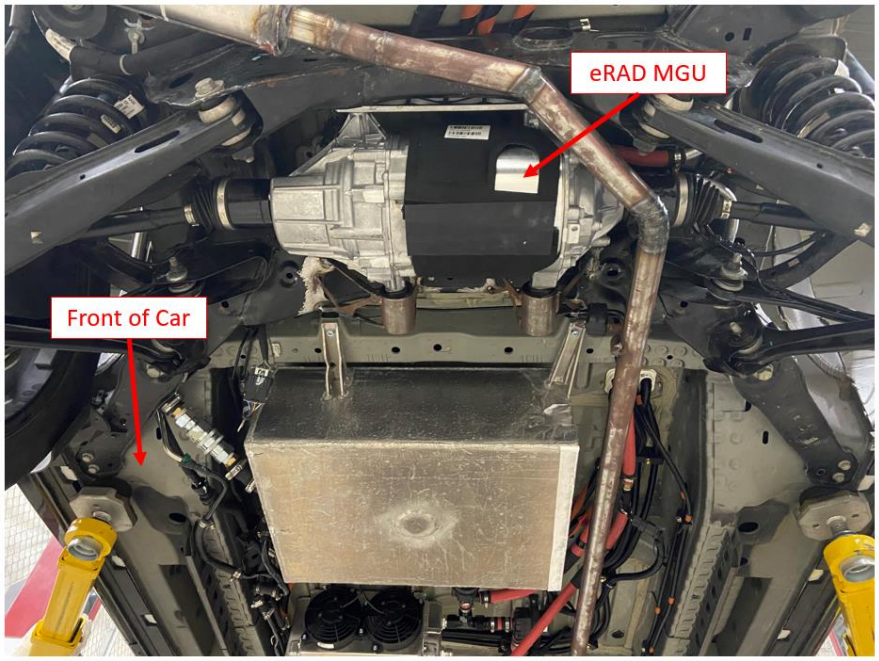

Figure 9: Underside of vehicle showing Magna eRAD MGU packaging in rear subframe

The Blazer differential mounted via two bushings in the rear section of the rear subframe, which are shown in Figure 10. A third bushing was attached to the driveshaft. The Magna eRAD was designed for four mounting points, two at the rear of the MGU and two at the front. The two rear eRAD mounting points did not locate to the OEM differential bushings and the front subframe had no mounts for the front of the motor. As such a rear MGU mount was engineered to adapt to the rear bushings and a front mount was engineered to mount to the front of the rear subframe.

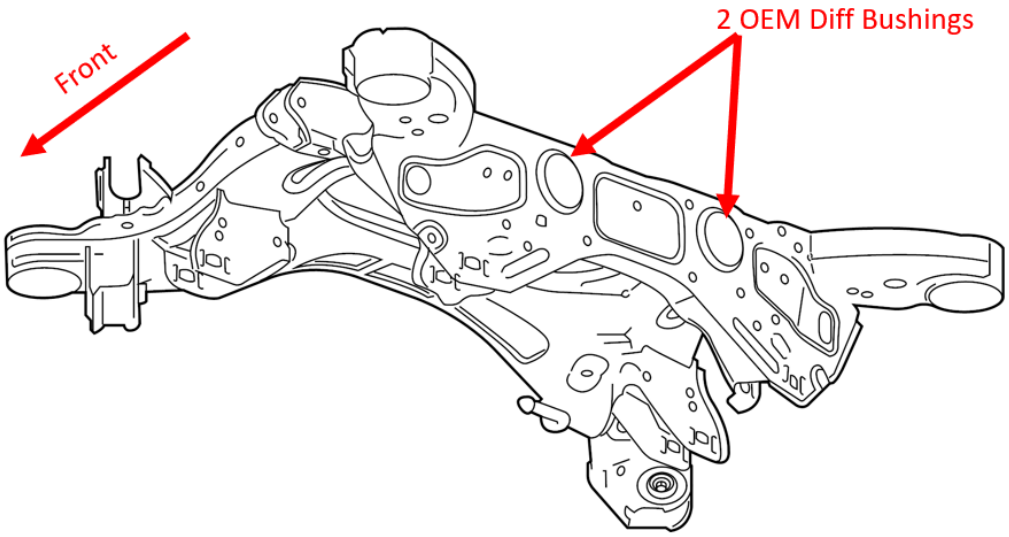

Figure 10: OEM Blazer rear subframe [5]

The front mount went through several design iterations. Initial designs focused on bolting to a convenient OEM hole in the middle of the subframe. The designs utilized a single bushing and mounted to the front two holes on the MGU. This would allow for servicing of the mount and did not require modification to the subframe. An example of one of the later iterations of this concept is shown in Figure 11. However, iterations on this concept introduced high stress to the subframe, since the subframe had little support for bending around this hole. Furthermore, the stresses in the mount arms were too high for the post welding, annealed metal.

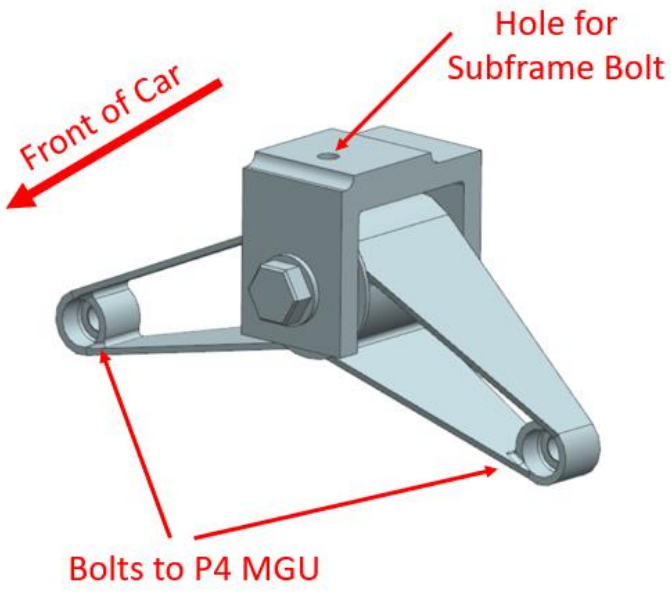

Figure 11: Early iteration on P4 MGU front mount

Later designs of the part switched to a two bushing mount that was welded in two parts to the subframe. Welding two mounts to the subframe facilitated a stronger connection in comparison to the single bolt in the middle of the subframe. This improvement comes at the cost of a permanent connection to the subframe. Communication was also required with GM to ensure that the welded connection would not compromise the strength of the subframe.

The final front mount design incorporates a boxed structure that can be easily made from waterjet plates and welded together. 4130 steel tubing is used as bushing cups. 0.125" thick 4130 plate is used for the box faces. Dorman 523-223 bushing were selected for the front mounts due to their small size and weight. The bushings are pressed in towards the front of the car. Combined with the rear bushings being pressed in towards the rear of the car, this prevents the bushings from sliding out during a crash. The edges of the mount stretch out towards the subframe to connect at stiff locations. This reduces the peak stresses

found in the subframe. The design underwent topology optimization to remove unnecessary weight from these plates. The design before topology optimization is shown in Figure 12. The upper portion of these mounts is contoured to match the bottom of the rear subframe.

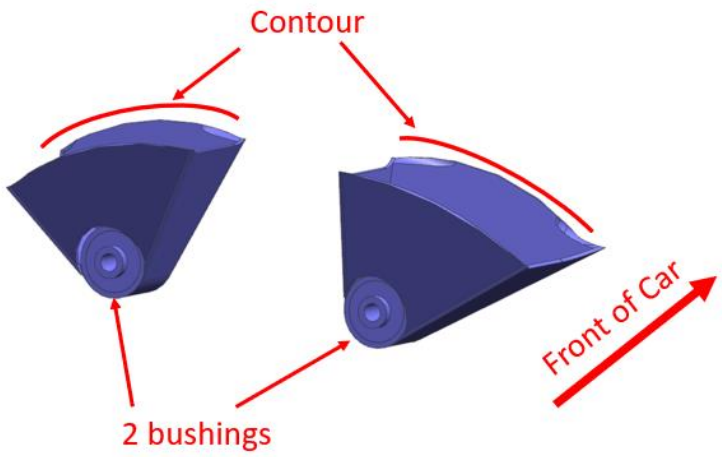

Figure 12: P4 MGU front mounts before topology optimization

The rear mount needed fewer iterations. The challenge of the design involved adapting from the OEM bushings to the rear eRAD mounting points; the eRAD mounting points are very close to the bushing bolt holes. Furthermore, the part could not be made thicker because the eRAD was packaged close to the rear bushings. The final design for the rear mount is shown in Figure 13. The mount is water jet and then CNC milled out of aluminum 7075-T6. The upper holes are tapped, and bolts are threaded through the rear bushings into the rear mount. The spacer on the upper holes facilitates a gap between the rear mount and the subframe. The lower holes utilize recessed screws to fit a bolt head between the subframe and the rear mount. Most of the load on the adapter plate is transferred from the lower bolt to the upper bolt so the plate could theoretically be made in

two parts. The plate was made in one piece to be conservative with stresses and improve serviceability.

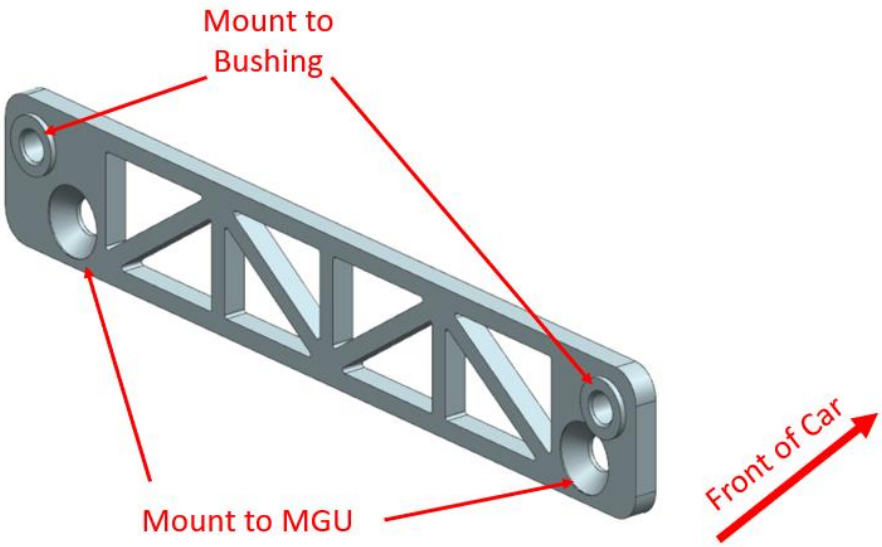

Figure 13: P4 MGU rear mount final design

3.3.2 Analysis

Forces on the four Grade 10.9 M12 bolts holding the motor were calculated for a 20 G lateral crash. The acceleration forces of the motor were assumed to act in a shear force shared equally between the four bolts. The FOS for yield in this load case is omitted for confidentiality but is much higher than 1.5.

A P4 mount finite element analysis (FEA) model was developed in HyperMesh to analyze the stresses in the mount design and remove weight in the front mount. The subframe and front mounts were modeled with 2D shell elements. The bushings, rear mount, and front bushing cups were modeled with 3D elements. The MGU was modeled as a point mass rigidly connected to the bushings. Bolts were modeled as 1D steel columns.

The model was supported at the subframe sides. The load cases used were full MGU torque, 20 g accel in the X axis, 20 g acceleration in the Y axis, and 8 g acceleration in the Z axis. The negative force versions of these load cases were also included (i.e. – 20 g acceleration in the X axis). The model is shown in **Figure 14** with the subframe omitted to protect confidentiality. Stress results for the model are shown in Table 4.

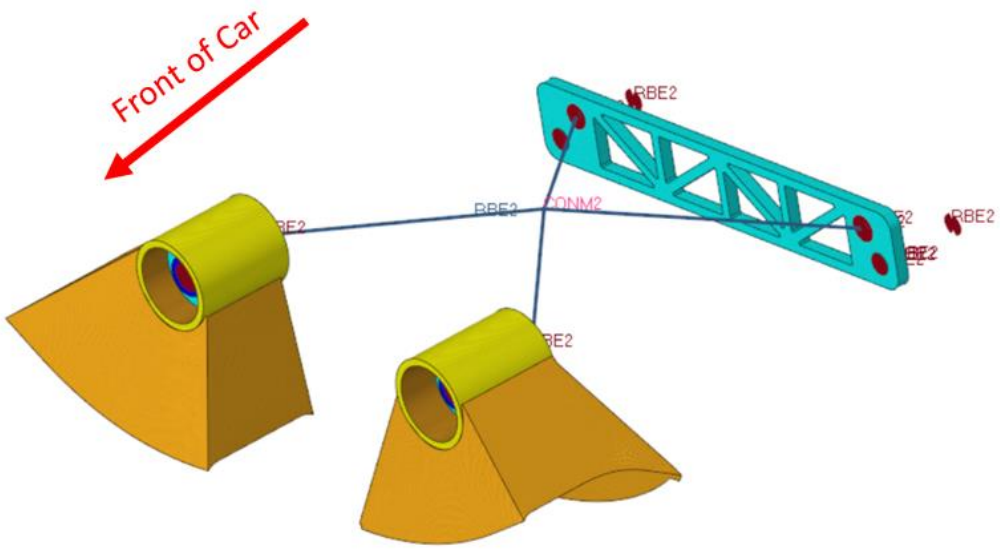

Figure 14: P4 MGU mount mesh in HyperMesh, subframe omitted

Table 4: Material properties and FOS for final, pre topology optimization P4 MGU mount

	Yield Stress [MPa]
Front Mount (4130-O)	360
Frame (GM proprietary steel)	—
Welds (ER70s-2)	420
Rear Mount (7075 Al)	500

	Stress [MPa] (Factor of Safety)			
	Front Mount	Frame	Welds	Rear Mount
MGU Torque	176 (2.0)	(1.2)	133 (3.2)	253 (2.0)
20 g X	152 (2.4)	(2.3)	53 (7.9)	56 (8.9)
20 g Y	82 (4.4)	(4.1)	75 (5.6)	117 (4.3)
8 g Z	47 (7.7)	(4.6)	36 (11.7)	48 (10.4)

Topology optimization was performed on the front mount. Mass was minimized with Von Mises stress constraints for each component. A cluster average of 1 was used for the stress constraint. The max allowed front mount stress was 200 MPa, the welds 165 MPa, and the frame 160 MPa in order to give conservative factors of safety. The optimized density plot is shown below in Figure 15, where 1.0 is an element with normal density. The plot uses an isosurface to filter elements with densities less than 0.1.

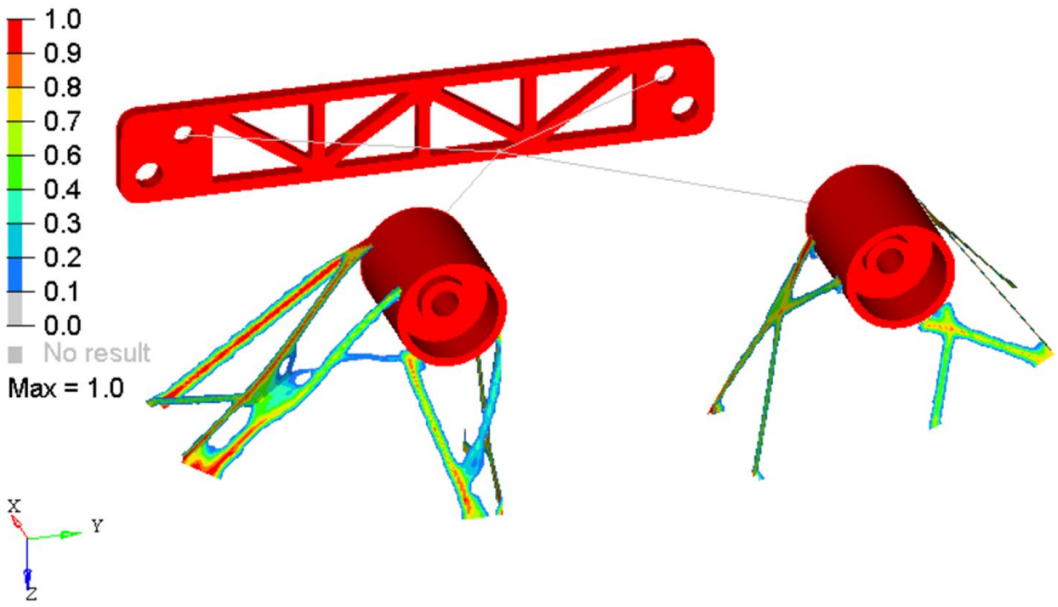

Figure 15: P4 MGU front mount topology optimization normalized density plot in HyperView

The optimization results informed a new computer aided design (CAD) model with cutouts which was then reinput into the FEA model. Stress and buckling verification studies were ran. For the buckling analysis, 1D RBE2 and RBE3 elements were investigated to see if they were more accurate than the solid modeled bushings. KGRGD was set to yes for the buckling analysis. The RBE3 elements gave anomalously low buckling safety factors, the solid elements gave believable safety factors, and the RBE2 element gave slightly higher safety factors. It was decided that solid elements gave the most realistic results. Altair was consulted to verify the credibility of the process. Buckling Factors of safety are shown in Table 5, where negative values mean the force must be reversed to see buckling. If the absolute value of the FOS is less than one, then the simulation predicts buckling. The factors of safety in this analysis inspire confidence that they will not buckle.

Table 5: P4 MGU mount buckling FOS

Load Case	Buckling FOS
MGU Torque	-9.9
20G in X	5.9
20G in Y	20.7
8G in Z	32.4

Stress is shown for each of the load cases in Figure 16. The plot shows Von Mises stress with a 1.5 multiplier. Peak Von Mises stress for negative loadings on each subcase (i.e. -20G_Y) is identical to its positive counterpart.

Table 6 summarizes the material properties and peak stresses for this model. Factors of safety are lower than their pre-topology optimization stresses for every part except MGU Torque loading of the frame. Here, the topology optimization reduced a stress concentration caused by the front mount coming to a sharp edge at the frame. Factors of safety are very similar between the two runs for the rear mount since it was not changed.

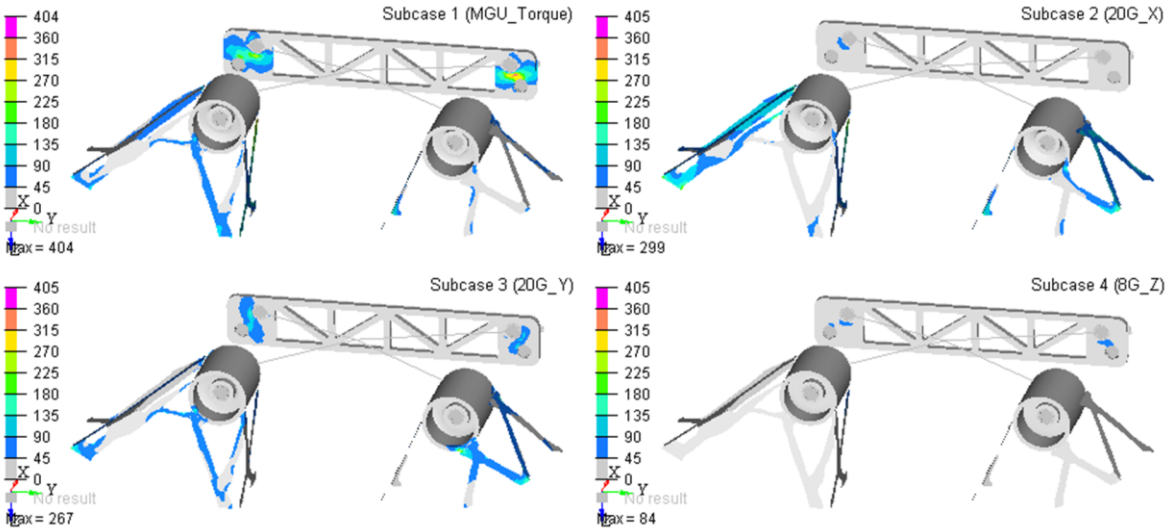

Figure 16: Von Mises stress contour [MPa] with 1.5 multiplier for P4 MGU mount verification study

Table 6: Material properties and FOS for final, topology optimized P4 MGU mount

	Yield Stress [MPa]
Front Mount (4130-O)	360
Frame (GM proprietary steel)	___
Welds (ER70s-2)	420
Rear Mount (7075 Al)	500

	Stress [MPa] (Factor of Safety)			
	Front Mount	Frame	Welds	Rear Mount
MGU Torque	209 (1.7)	(1.7)	215 (2.0)	269 (1.9)
20 g X	199 (1.7)	(1.9)	190 (2.2)	56 (8.9)
20 g Y	147 (2.4)	(1.9)	172 (2.4)	117 (4.2)
8 g Z	47 (7.7)	(7.3)	56 (7.5)	51 (9.8)

The initial design weighed 2.65 kg including the plates and cups. The optimized design weighs 1.15 kg resulting in a 1.5 kg decrease in weight. Although this weight reduction is not significant in terms of the overall weight of the car, the techniques used came at minimal extra effort during the manufacturing process and much was learned about stress and buckling.

3.3.3 Fatigue

Fatigue was a concern for the P4 mounts, given that the P4 motor will alternate between motoring and generating many times over a drive cycle. The vehicle is not intended to see similar mileage accumulation to a production vehicle due to the limited competition length of four years, with less than two of them including a driving car. However, Over the life of the vehicle the mount should still be able to withstand at least 5,000 vehicle miles, as estimated from the mileage accumulation of previous competition vehicles.

The endurance limit of the 4130-O front mounts was calculated using

$$S_e' = 0.5 S_{ut} \tag{11}$$

$$S_e = k_a k_e S_e' \tag{12}$$

to be 114 MPa. The ultimate tensile strength of 4130-O was taken to be 560 MPa. k_a and k_e were estimated from *Shigley's Mechanical Engineering Design* to be 0.501 and 0.814, representing a forged surface finish and a 99% reliability [6]. The front mount edges have a waterjet cut surface finish and these rough edges were conservatively approximated with a forged surface finish. Number of cycles to failure, *N*, was calculated using

$$a = \frac{(fS_{ut})^2}{S_e} \tag{13}$$

$$b = -\frac{1}{3}\log\left(\frac{fS_{ut}}{S_e}\right) \tag{14}$$

$$N = \left(\frac{\sigma_a}{a}\right)^{\frac{1}{b}} \tag{15}$$

where f was estimated from Figure 6-18 of *Shigley's Mechanical Engineering Design* to be 0.875 [6] and σ_a was taken from the FEA to be 209 MPa at full MGU torque. This gives an estimated cycle count of 57,000 cycles. Similar calculations were performed for the 7075-T6 rear P4 mount using an ultimate tensile strength of 572 MPa and an endurance strength of 159 MPa, resulting in an estimated cycle count of 16,500 cycles.

Miner's rule was used to calculate the damage the P4 motor mount would experience in the EMC city drive cycle shown in Figure 17. The top graph is the motor torque the P4 motor would need to supply over the drive cycle if it is the sole supplier of torque and there is no onboard engine. The bottom graph is velocity over the EMC city drive cycle. For calculating fatigue cycles, motor torque is assumed to be cyclic and fully reversed. This correlates with the drive cycle torque, where every strong acceleration event is followed by a similar regen event. The many small deviations in motor torque have small amplitudes and were assumed to be below the endurance limit of the mount. S_e was calculated for the rear mount to be 109 MPa. This stress is calculated to occur at 40% MGU torque so red lines were drawn on the plot to measure dips below this level. σ_a for each cycle in the Miner's rule sum was calculated by

$$\sigma_a = abs(Motor\ Throttle) * \sigma_{FEA} \quad (16)$$

where σ_{FEA} is the peak stress found in the component during the MGU Torque load case in the FEA. The peak stress in the rear P4 mount is in tension during regen, so the measured cycles were from the regen peaks. Dips below -100% throttle were clipped at -100% throttle.

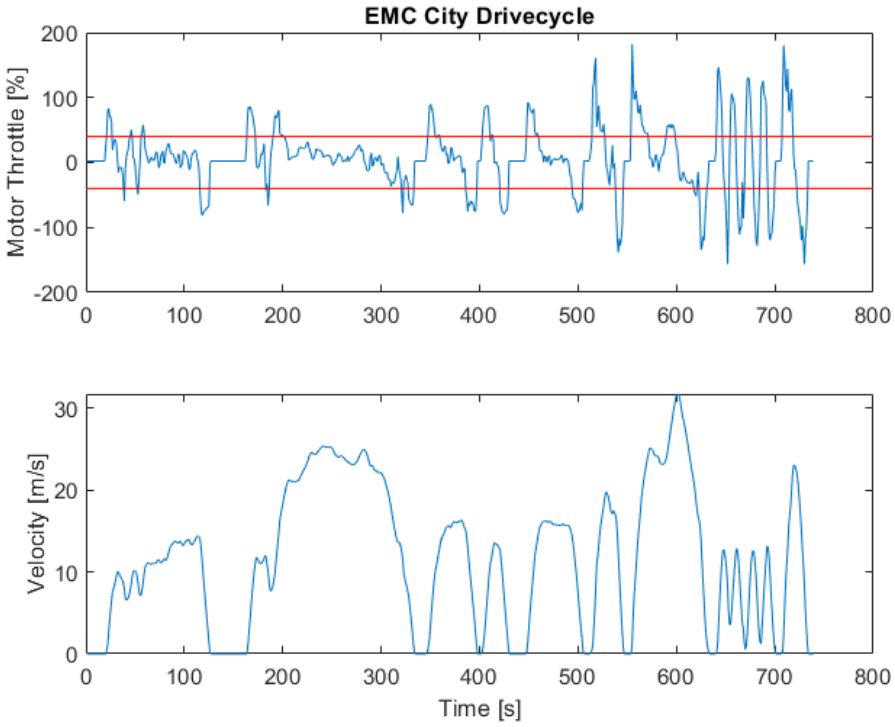

Figure 17: EMC city drive cycle if P4 motor is the only source of traction.

The rear mount is estimated to withstand 1847 EMC city drive cycles. Given that the cycle is 8.6 km long, the rear mount has a range of 15,800 km or 9,800 miles. The front mount is estimated to withstand a longer 55,000 km.

The number of torque cycles the P4 motor would experience was also estimated by analyzing the torque requests from dynamic programming with the chosen P0P4

architecture, shown in orange in Figure 18. Miner's rule was similarly applied. The rear mount is estimated to withstand 6634 cycles for a range of 54,000 km.

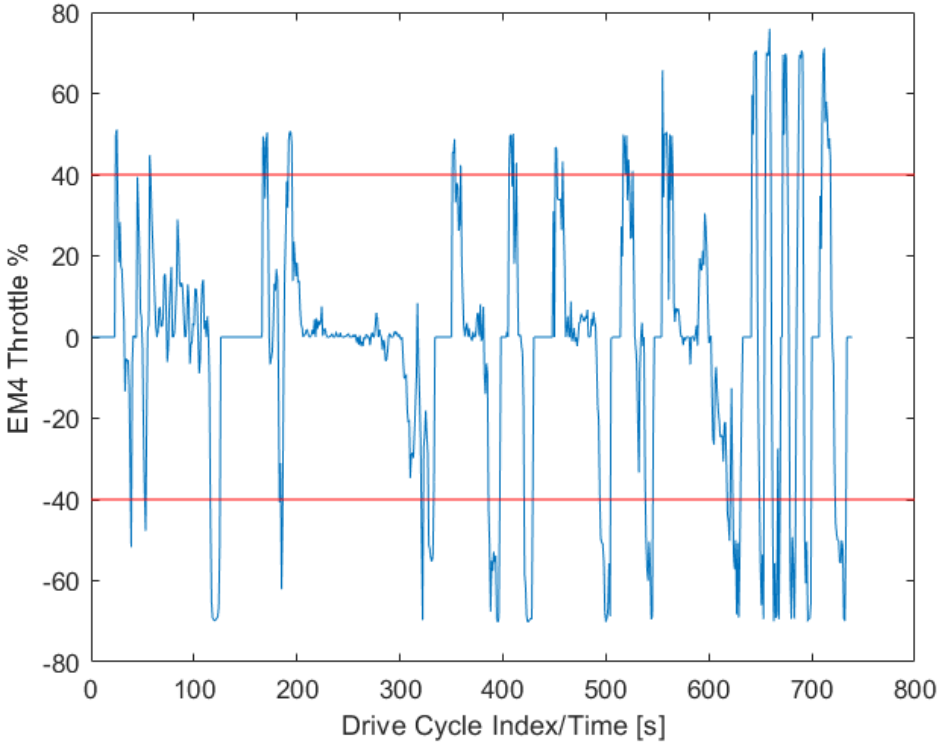

Figure 18: Dynamic programing derived P0P4 control for EMC city drive cycle, showing only P4 MGU throttle

Fatigue is less for the P0P4 drive cycle because part of the regen braking is carried by the P0 MGU. In the GT Blazer, the P0 is not expected to contribute significantly to regen braking. Thus, fatigue life is likely closer to that of the P4 only drive cycle. The predicted range of the rear mount would be unacceptable for a production vehicle, which would drive for hundreds of thousands of miles. However, as the worst case 9,800 mile predicted lifetime is substantially greater than the GT Blazer's expected lifetime of 5,000

miles, fatigue should not be a significant concern. Furthermore, the EMC city drive cycle contains many aggressive accelerations and decelerations within a short distance. In contrast, the EMC highway drive cycle only sees 3 torque peaks over the same time frame while traveling a greater distance. The target market of the vehicle acquires around 50% of vehicle miles from city driving and 50% from highway driving, per the EcoCAR rules, so the fatigue estimates based on the EMC city drive cycle are conservative.

As the vehicle nears its end of life, the rear mount should be regularly inspected for cracks. If the P4 MGU mounts were to be redesigned, a steel rear mount should be considered to increase the part's life. Furthermore, the FEA boundary conditions around the bolted joints in the rear mount should be reconfigured with bolts instead of rigid elements to more accurately reflect the connection. If the GT Blazer were to be manufactured in large quantities, such as in a production run, the analysis should be recomputed with a k_e that reflects the larger run to minimize the chances of a mount fatiguing within warranty. Lastly, analysis of fatigue life should be done before a stress constrained topology optimization is performed to ensure any compromise between weight, stiffness, and cycle life is quantified and acceptable. The above P4 MGU mount design was optimized before a fatigue study was performed, which could have resulted in a lower fatigue life than needed. Due to the rear mount not changing and it being the first to fail, both the optimized and unoptimized designs have similar fatigue lives.

3.3.4 Manufacturing

One side of the front mount as manufactured is shown in

Figure 19. Small legs were added to the design and were then cut off with an angle grinder. These legs helped to locate the four corners of each waterjet plate during welding. After these legs were cut off, the front and rear mount were bolted to the Magna eRAD and raised into the subframe. The rear bushings were attached to the rear mount and the front mounts were adjusted to touch the subframe. Lastly, the front mounts were welded to the subframe. The final mounts are shown before they were painted in

Figure 20.

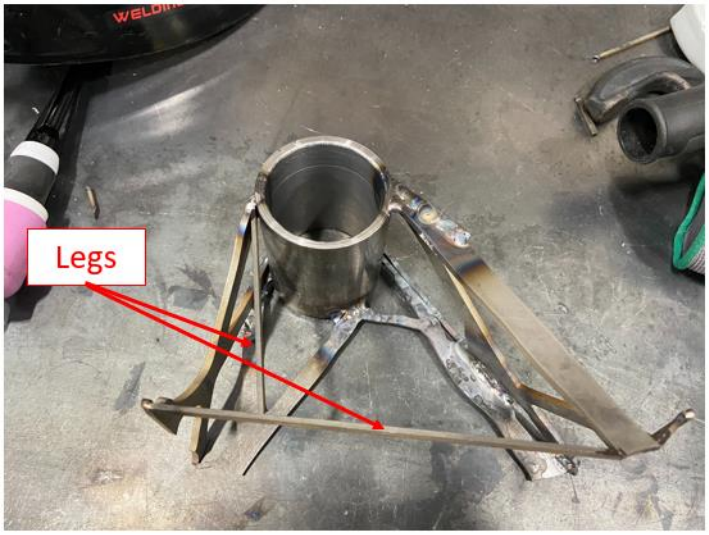

Figure 19: P4 MGU front mount, one side post welding

Figure 20: P4 MGU front mounts as installed on car

3.4 Fuel Tank

3.4.1 Design

A non-OEM fuel tank is required by the EMC rules. The teams are required to remove and install their fuel tanks within 30 minutes. The fuel tanks also have a weight limit of 100 lbs. including fuel, fuel pumps, lines, and any other equipment that is removed with the tank. These rules make servicing fuel tanks easier and encourage the use of compact, lightweight designs to get the best vehicle range between refueling stops. A high cruising range is something the GT EcoCAR team's target market desires, so the team spent effort making the fuel tank light with a high capacity.

A custom fuel tank which could easily package under the car was investigated. Custom fuel tanks are required by the EMC rules to be either steel or aluminum. Steel was avoided due to its tendency to rust. The material chosen was Aluminum 6061-T6 for its

balance of specific strength and manufacturability. The team designed the tank to be manufactured via in house waterjet and welding. The tank was heat treated back to 6061-T6 after welding because welding the 6061-T6 annealed the metal in the heat affected zone (HAZ).

Other requirements of fuel tanks include a 1.5 FOS for 20 g in the X, 20 g in the Y, and 8 g in the Z acceleration loads. The tank must be pressure tested to be leak free at 6 psi gage. The tank must also be protected by the vehicle frame from hitting the ground.

A picture of the final design is shown in Figure 21. The tank is made of waterjet aluminum plates that have been welded together. The tank uses the OEM Blazer fuel pump. It also has a fill tube that attaches to the OEM Blazer fill line, a 2.5 psi pressure vent, and two ports for fueling and venting outside of the car. The tank bolts to the Blazer in three places. Two threaded holes were used from the OEM fuel tank mounts and the last was an existing hole in the frame the GT team welded a weld nut to. The attached fill line, fuel sender line, and evaporative line all have quick disconnect fittings to facilitate fast service. Dropping the tank only requires removing three bolts, one electrical connector, and three quick disconnects. The tank can be raised and lowered using a transmission jack.

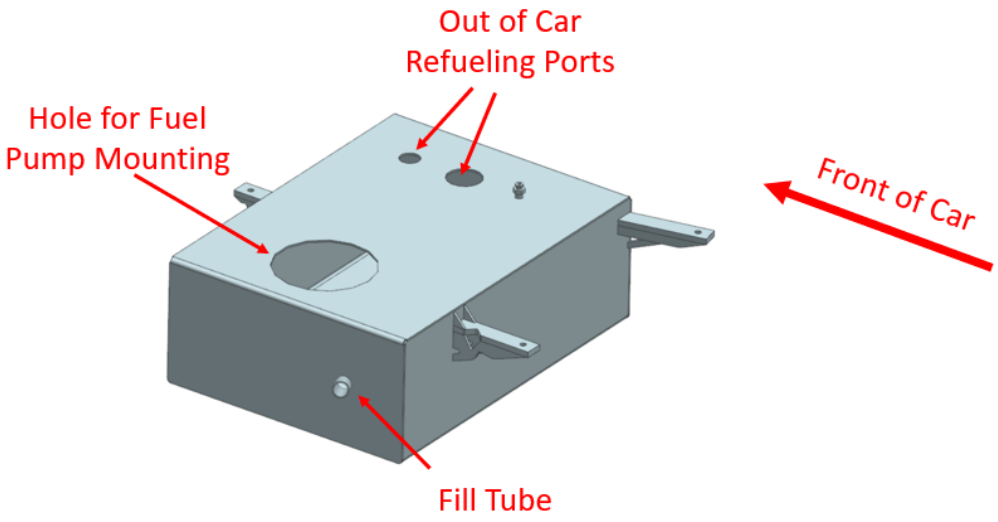

Figure 21: GT fuel tank CAD

Several iterations were performed on the out of car venting and sealing bungs. The first two designs utilized uncoated and anodized NPT bungs. These never sealed, even with PTFE tape, and eventually galled. The final solution used anodized aluminum AN O-ring bungs. These seal with O-rings instead of thread deformation and are therefore much more resistant to galling.

3.4.2 Analysis and Modification

Forces on the three Grade 8.8 M8 bolts fastening the tank were calculated for a 20 G lateral crash, assuming the full tank weighs 100 lbs. The acceleration forces of the fuel tank were assumed to act in a shear force shared equally between the three bolts. The FOS for

yield in this load case is 4.1. This FOS should be revisited after determining the weight of the tank when it is filled with fuel, as would be done at a gas station.

A finite element analysis was performed on the tank using HyperMesh. The tank was modeled using 2D elements of correct thickness for the manufactured plates and tabs. Welds were modeled at 3.175 mm thickness. The mass of fuel in the tank was modeled using a 1D point mass at the center of the tank. The point mass is connected to the tank walls using an RBE3 element. This does not accurately predict stress on the tank walls due to hydrostatic pressure but does resolve stress in the tank mounts. The tank was loaded with separate 20 g in the X, 20 g in the Y, and 8 g in the Z acceleration load cases. Ribs were added to the mounting tabs until the tank passed with a 1.5 FOS. Next, topology optimization was performed on the tabs and ribs to remove unnecessary weight and therefore increase fuel capacity. The tank was optimized for minimal mass with a stress constraint of 180 MPa. A density plot of the topology optimization results is shown in Figure 22.

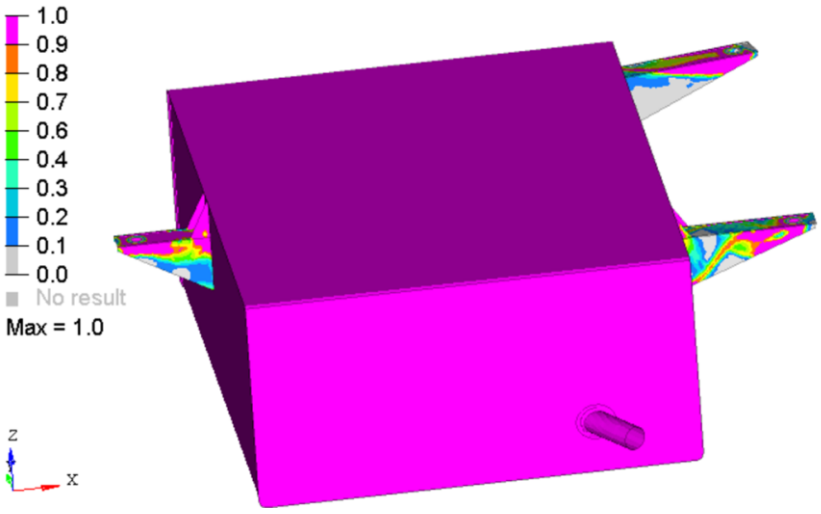

Figure 22: Fuel tank topology optimization normalized density results

These shapes were realized in CAD by making conservative cutouts in places with less than 0.1 density. This model passed a verification FEA and was manufactured. However, the 6 psi gage pressure load case was neglected in the analysis. During pressure testing, the tank was found to balloon and yield due to its large unsupported top and bottom faces. A postmortem FEA showed yielding at 4.3 psi. A stress plot with the yielding parts of the tank in red is shown in Figure 23. Revisions had to be made to the tank for it to safely be pressure tested to 6 psi. A 0.12” thick, 2” OD 6061-T6 column was welded into the center of the tank to hold the top and bottom plates in tension during the pressure test. Small holes were drilled in the top and bottom of the column to allow fuel to enter the column and this is shown in Figure 24. This facilitates a negligible reduction in tank fluid volume. The column as installed is shown in Figure 25. A cap was then welded over the column to seal the tank.

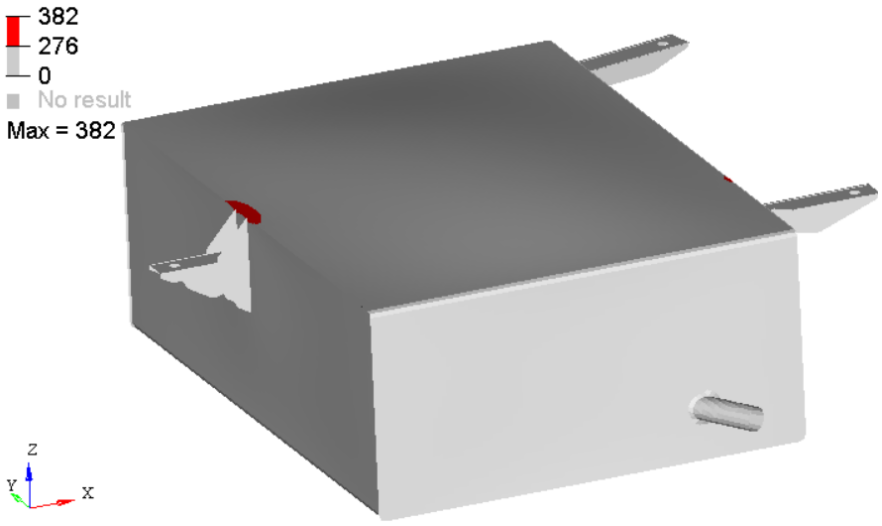

Figure 23: Fuel tank Von Mises Stress contour [MPa] showing yielding for 6 psi load case

Figure 24: Fuel tank column with holes drilled at ends

Figure 25: Column installed in the fuel tank, cap to be welded later

A final FEA was performed on the tank with the column insert and the 6 psi load case. Table 7 shows the results of this study and FOS for each load case. Peak Von Mises stress for each load case is shown in Figure 26. Peak stress is at the bolted connection in one of the tabs for the 20 g X, and 8 g Z load cases. Peak stress is at the weld of a rib for the 20 g Y load case. During manufacturing, a focus was placed on minimizing stress concentrations for this weld with a smooth toe blend angle. Peak stress for the 6 psi load case is on the weld of the column. Note that the cap has been hidden in this view to show the peak stress. Stress concentrations at this joint were also minimized during welding.

Table 7: Fuel tank verification FEA FOS

	Yield Strength:	276	MPa
Loadcase	Peak Stress [MPA]		FOS
20G_X	159		1.7
-20G_X	159		1.7
20G_Y	166		1.7
-20G_Y	166		1.7
8G_Z	117		2.4
-8G_Z	117		2.4
6PSI	169		1.6

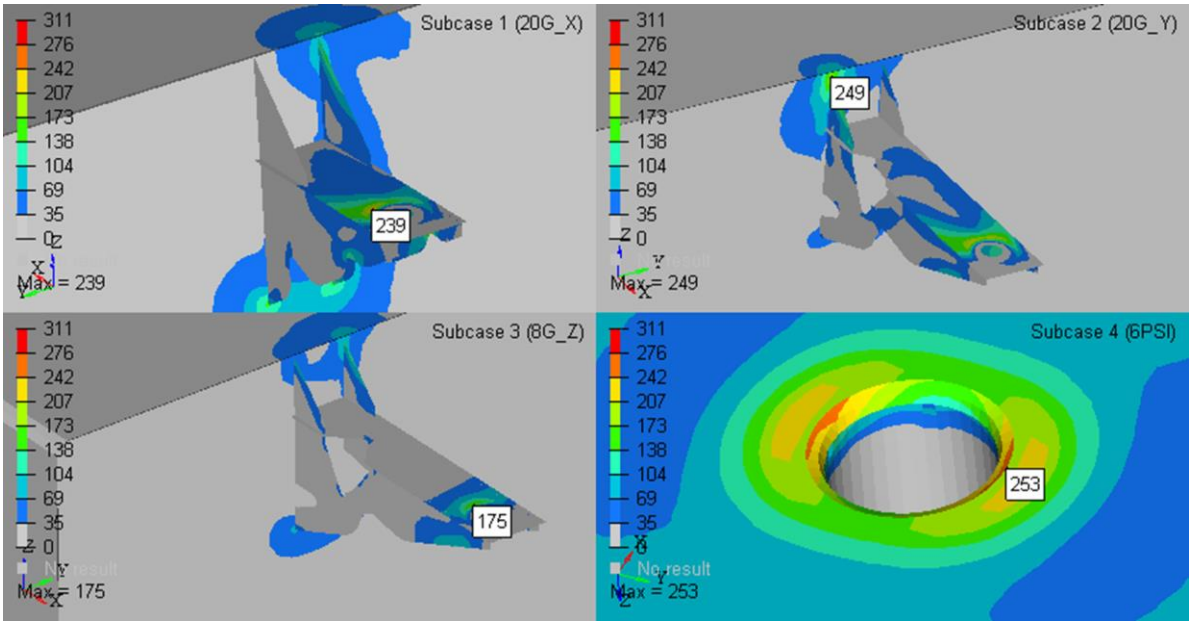

Figure 26: Von Mises stress [MPa] with 1.5 multiplier for 20 g in X (top left), 20 g in Y (top right), 8 g in Z (bottom left), 6 psi (bottom right)

3.4.3 Conclusion

After installing the column, the tank was pressure tested to 3 psi to find any leaks. These were filled, then the tank was taken to Aerospace Fabrications of Georgia who graciously sponsored the GT team by heat treating the tank to 6061-T6.

The tank weighs 25.3 lbs. allowing for 10 gallons of fuel and, using the Simulink ECMS model fuel economy, an estimated range of 327 miles. The tank can be removed and then installed by two students in less than 6 minutes. A picture of the finished tank is shown below in Figure 27.

Figure 27: Finished fuel tank

3.5 A/C Compressor Mount

3.5.1 Design

Due to the simplification of the LCV belt drive system and the availability of HV power, the GT team opted to use a Denso ES34 HV A/C compressor. A custom mount was engineered to adapt this compressor to the car. Initial packaging studies tried to place the compressor on the side of the engine with an adapter plate. The HV compressor produces a lot of vibration that could cause issues if it were directly mounted to the chassis. If the compressor were to be mounted to the engine, the compressor vibrations would be mitigated by the engine mounts. No room was available on the front side of the engine, so space on the backside was explored. Unfortunately, the compressor would not fit between the intake manifold and frame of the car. Furthermore, the A/C compressor has a limited range on axial inclination and rotation angle, so orientating the compressor a creative way would not work. The location decided on for packaging is the space in the top left corner of the engine bay, recessed by the fire wall and the frame rail. This location is only occupied by A/C lines on the stock vehicle so minimal modification is required. However, this necessitates mounting to the chassis and an increased focus on minimizing the compressor vibrations transferred to the chassis. During engine start up there is sufficient clearance between the compressor and the engine for the engine's cranking movement. Figure 28 shows the A/C compressor location in the engine bay and gives a sense of its clearance to the engine.

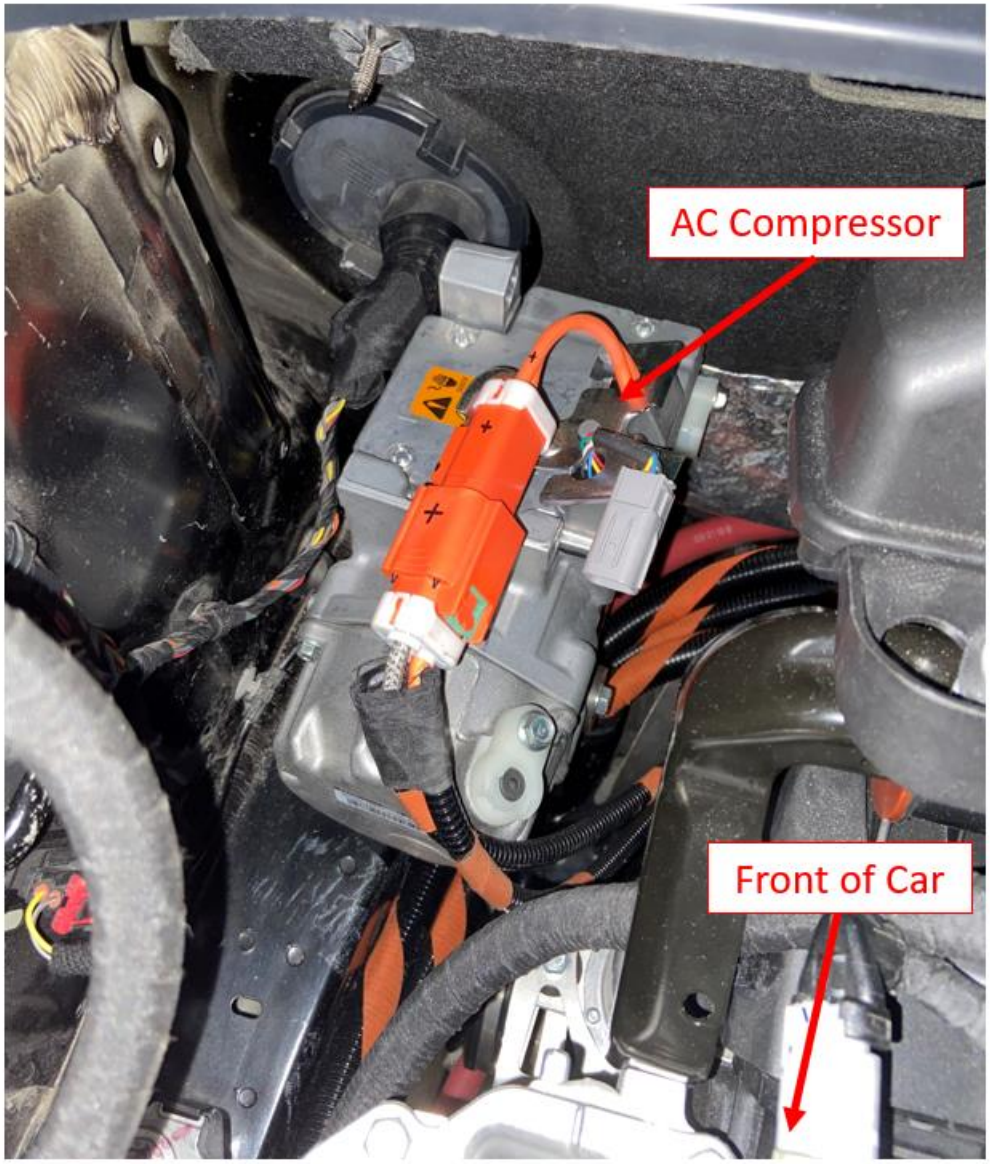

Figure 28: HV A/C compressor as mounted in car

The final compressor mount design utilizes HA King bonded bushings and snubber mounts to reduce the transfer of compressor vibrations to the chassis. Finding a bushing that could withstand the 20 g X, 20 g Y, and 8 g Z acceleration loads was difficult. Additionally, these bushings are typically made in single run quantities larger than the GT EcoCAR team could afford. The bushings selected were bushings that the supplier

happened to have in stock and graciously agreed to sell in low quantity, the 5007-03 bushing. This bushing meets normal driving loads but will likely fail in a crash. During a crash, the bushings will fail and the steel bolts holding the assembly together will be captured by the HV A/C compressor mount, which is designed to not yield in a crash. The calculations supporting these arguments are shown in the next section.

The final design is shown in Figure 29. It is bolted to the chassis via weld nuts attached to the frame. This bolted design facilitates an easier redesign if different bushings need to be sourced. The mount is made in 3 parts out of 0.125" thick 4130-O plate. All parts were cut on a waterjet, bent to precise angles on a sheet metal bender, and welded together. 4 HA King 5007-03 bushings isolate the compressor vibrations. Grade 8.8 M8 bolts fasten the A/C compressor to the mounts. Grade 8.8 M6 bolts bolt the mount to the chassis.

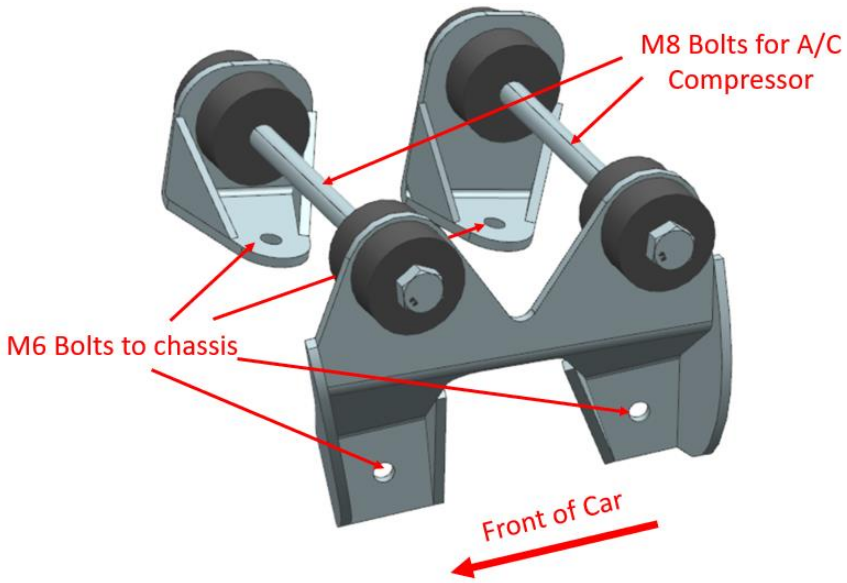

Figure 29: HV A/C compressor mount CAD

3.5.2 Analysis

The load cases considered are a 20 g X, 20 g Y, and 8 g Z acceleration. A 1.5 FOS is desired. The bushings are rated to 458 N axially and 160 lbs. radially. The axis of the mounted bushings aligns with the Y axis of the vehicle. The force on one bushing was calculated as

$$F = ma * \frac{1}{4} \tag{17}$$

where the force, *F*, is assumed to be equally divided by the 4 bushings. Results of these calculations are omitted to protect the mass value of the compressor. The bushings would likely survive a 20 g axial crash but would certainly fail in a 20 g radial crash. Furthermore, to have a FOS of 1.5 in a 20 g crash, the bushings need a higher axial load rating. Under normal driving conditions, the worst acceleration a bushing might see could be a 4 g bump when accidentally hitting a curb. In these conditions, the bushings will not fail.

Shear force on each M6 bolt is the same as would be seen by a single bushing. Here, the weakest link is the M6 bolts holding the mount to the car. The tensile yield strength of an 8.8 M6 bolt is around 577 MPa. In 20 g radial crash, the bolt would have a high FOS. The FOS in shear for the M8 bolts that attach the compressor to the mount are even higher.

Topology optimization was used in deriving the shape of the mount. A packaging envelope model was created in HyperMesh to determine an efficient shape for the mount. Clearances were allocated for access to the bolts on the side of the mount. Mass was minimized with a Von Mises stress constraint. The model and results of this study are shown in Figure 30. The study did not reveal an efficient connection between the bolts and bushings on the right side of the mount, so liberty was taken to focus on simple

manufacturing for this connection. A new CAD model was created with bent, waterjet cut metal. Ribs were added until the model passed the load cases with a minimum 1.5 FOS. The final HyperMesh model is fixed at each of the 4 mounting bolt holes with constrained RBE2s. The bushings are modeled with RBE3s with a 1D mass element at the approximate CG of the compressor. The second iteration of topology optimization is shown in Figure 31, where parts of the added ribs were recommended to be removed.

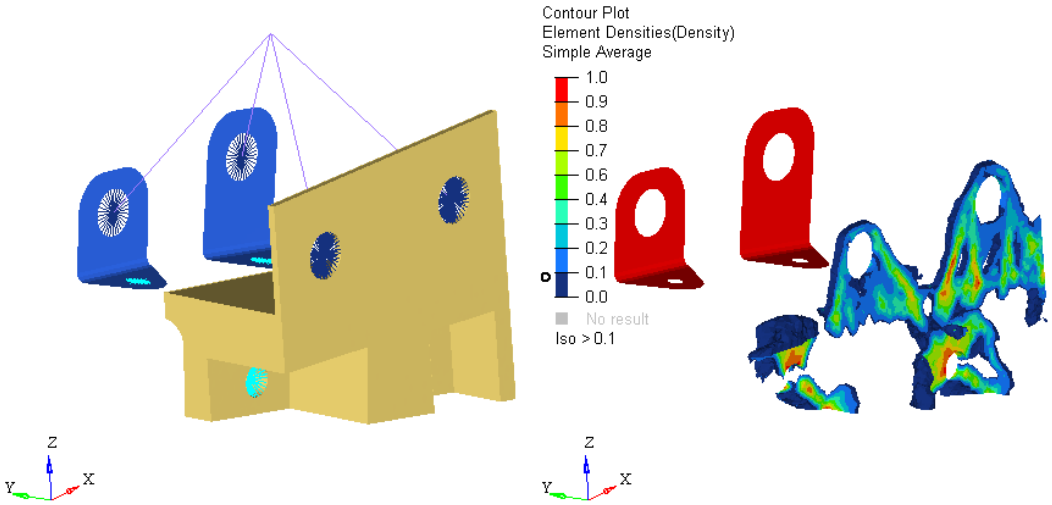

Figure 30: Packaging envelope for compressor mount topology study (left) and normalized density plot of optimization study (right)

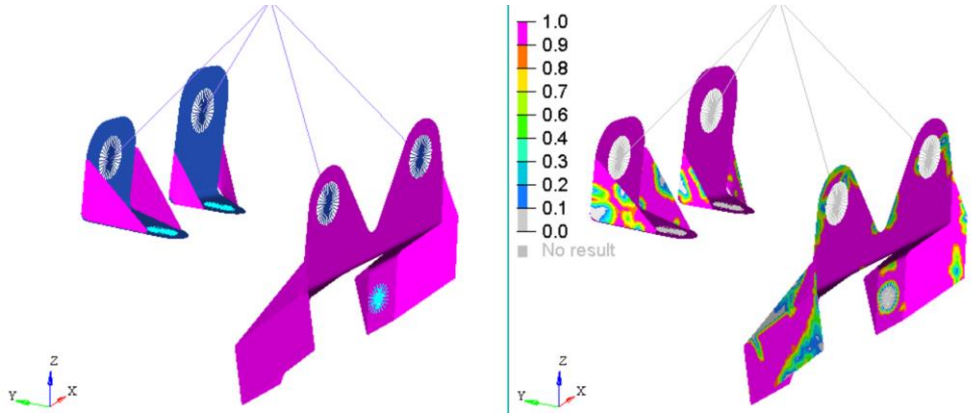

Figure 31: Second topology optimization design space (left) and normalized density plot of optimization (right)

A verification run was done after translating these results into CAD. An envelope load case showing the max stresses from the 20 g X, 20 g Y, and 8 g Z acceleration load cases is shown in Figure 32. A 1.5 multiplier is shown. The peak stress with a 1.5 multiplier is 402 MPa, which is higher than the yield strength of 4130-O, at 360 MPa. This is acceptable because stresses above yielding are all near the mounting bolts, which utilize rigid RBE2 elements and are likely conservative for stress results.

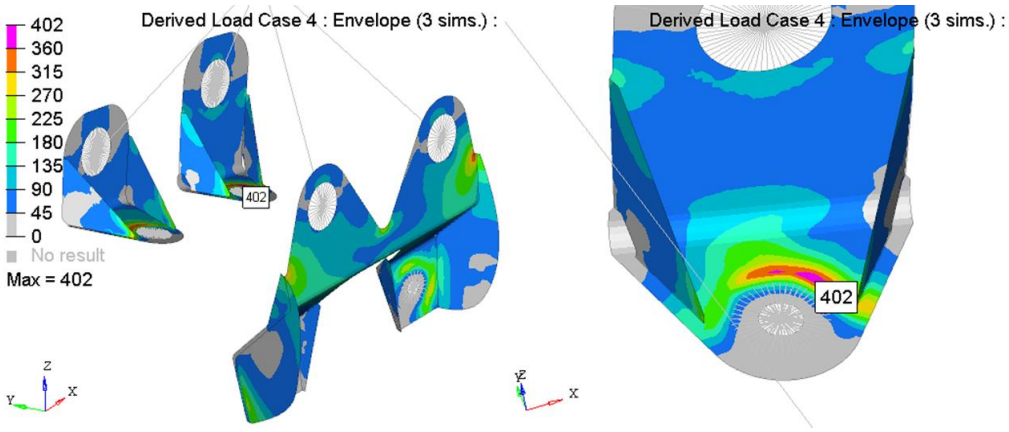

Figure 32: Von Mises stress [MPa] envelope load case with a 1.5 multiplier. Overview of result (left) and close up on peak stress (right)

3.5.3 Manufacturing

The holes in the chassis to were located by bolting together the mounts, bushing, and A/C compressor to use as a jig. The rubber bushings were not in hand during this time, so identical 3D printed models were substituted. The jig was placed over the chassis and the center of the holes marked. The holes were then drilled in the chassis and the weld nuts were stitch welded to minimize the impact of the HAZ on the chassis.

CONCLUSION

By the middle of year 3 (December 2020) of the competition, the GT EcoCAR team has successfully transformed the GT Blazer into a functioning hybrid with increasing autonomous function. At the time of writing, all vehicle components are integrated and all but the P4 motor are functional. Refinements are in progress to enable the vehicle to pass a technical inspection and the car will be tested on a closed course once this is completed. The choice to use the easy to integrate LCV power cube enabled a simple engine swap in year 2 of the competition. The choice of adding a P0 MGU to the P4 architecture enabled the team to run on a hybrid architecture if either of the motors stopped working. The HEV4 battery pack has been integrated without significant issue. More controls work needs to be done to increase the fuel economy of the architecture, but all components in the vehicle are functioning.

All powertrain components will be stress tested via evasive driving once the vehicle is sufficiently complete to drive on a closed course. It appears the P4 motor will have vibration issues because there is little space for movement between the rear mounts and the subframe. The GT team's fuel tank is successfully installed in the vehicle and facilitates a range suitable for the competition and is attractive to the target market. The HV A/C system has been integrated into the vehicle and facilitates comfortable cooling of the vehicle's occupants. The widespread use of topology optimization across several components on the vehicle saved some amount of weight. The biggest advantage of these techniques is that they gave the students a better understanding of stress progression

through a structure and an understanding of how optimization can be used to educate design.

The dynamic programming model is unlikely to be further integrated into the vehicle but its control decisions can be used as a benchmark for a successful implementation of on-vehicle ECMS. In future years of the competition, the use of dynamic programming as a control strategy could be explored. GPS navigation routes are commonly used in the MaaS market and dynamic programming derived control strategies could be calculated once the route is decided.

APPENDIX: DYNAMIC PROGRAMMING CODE

```
function
DP_Fuel_Consumption_EcoCAR_P1P4_vectorized_v2(engineFileName,stat
e_GRID_size,control_GRID_size,cycleFileName)
% Dynamic Programming HEV script
% Written by Michael J. Leamy, August, 2011
% Edited by Christian Free, January 2019
% Edited by Jason Calvert, Febuary 2019


close all
tic

%% inputs
% state_GRID_size = 500;
% control_GRID_size = 20;
alpha_GRID_size = 50;   % added additional control vector because
it to see if u of alpha takes more time. it's about the same, but
alpha doesn't help accuracy
% Recomend S = 10000, u = 500 for <1% error
%.832 for diesel, .745 for gasoline
FuelDensity = .745;   %kg/L
% Note: this uses no interpolation, and instead next nearest
neighbor for
% state

% cycleFileName = 'EMC_City';
%load Schedule_FU505_Ten_Times.mat
% load Schedule_FU505.mat;
load([cycleFileName '.mat'])
%load Schedule_Boston_Cab.mat;


% note that the .mat file must be dissassembeled because we are
unable to
% add variables in a nested function
% engineFileName = 'LYX15.mat';
EngineData = load(engineFileName);
Engine_Fuel_Data = EngineData.f_fuel .*1000; % This is not BSFC,
but instead grams/sec as a function of throttle request and
engine RPM
Engine_Fuel_RPM_Axis = EngineData.f_tbrake_n_bpt';
Engine_Fuel_Throttle_Axis = EngineData.f_tbrake_t_bpt ;     % for
this engine, commanded torque functions just like throttle
Engine_Torque_Data = EngineData.f_tbrake ;
Engine_Torque_RPM_Axis = EngineData.f_tbrake_n_bpt';
Engine_Torque_Throttle_Axis = EngineData.f_tbrake_t_bpt;
max_engine_RPM = Engine_Fuel_RPM_Axis(end);
```

```
omega_bpts = [];
Trq_bpts = [];
eta_tbl = [];
if engineFileName== "LGX36V6_V2.mat"
    load LGX36V6_trans
elseif engineFileName== "LCV25_V2.mat"
    load LCV25V2_trans
elseif engineFileName== "LTG20_V2.mat"
    load LTG20V2_trans
elseif engineFileName== "LYX15_V2.mat"
    load LYX15V2_trans
else
    warning("transmission data might not be valid")
end

% Vehicle data based on a 2019 Chevy Blazer
mass = [redacted];     % mass in kg no engine or transmission
motorMass = [redacted]; %[kg]    Parker 210-50=25, 210-100=35, 210-
150=46,
batteryMass = [redacted]; %[kg]
addMass = [redacted]; % additional mass, from model (inverter, fludis,
etc)
mass = mass+motorMass+batteryMass + addMass;
g = 9.81;        % accel of gravity in m/s^2
% Area = [redacted];    % frontal area in m^2 (guessed)
% rho_air = 1.25; % density of air in kg/m^3
% Cd = [redacted];       % drag coefficient
% fr = [redacted];      % rolling resistance coefficient
A = [redacted]; B = [redacted]; C = [redacted];    % vehicle road load equation
coastdown coefficients
delta_M = [redacted];  % mass factor guessed
radius = 0.36; % loaded tire radius, [m]
% eta = [redacted];      % transmission efficiency: This is only applied
to engine power, but the inefficiency will propegate into the
motor as well, so we don't need a motor term (I think)
accessoryLoad = 0.8; % [kW] includes all loads, (AC, CAVs,
headlights, etc.)

if strcmp(engineFileName,'LYX15.mat')  %M3U trans
    i_final = [redacted]; % Final drive ratio
    gear_ratios = [[redacted]];
    mass = mass+[redacted];
elseif strcmp(engineFileName,'LTG20.mat')    %M3H trans
    i_final = [redacted]; % Final drive ratio
    gear_ratios = [[redacted]];
    mass = mass+[redacted];
else % use trans values for LCV25, M3D trans
    i_final = [redacted]; % Final drive ratio
    gear_ratios = [[redacted]];
    mass = mass+[redacted];
end
```

```
[~,NUM_GEARS] = size(gear_ratios);

motorName = 'Denso-ISG';

Denso_eff = [];
Denso_rpm_axis = [];
Denso_torque_axis = [];
load 'DensoISG.mat';
EM_max_torque = [redacted];     % [Nm] Parker 210 motor
EM_max_power = [redacted]; % [Watts] max power of motor or battery,
whichever is limiting
pulleyRatio = [redacted];    % pulley ratio for BAS system
EM_gear_ratio = 1*pulleyRatio;    % Gear ratio between EM and the
final drive
EM_torque_min = -EM_max_torque;
EM_torque_max =  EM_max_torque; %u2
EM_power_max = EM_max_power;
EM_power_min = -EM_max_power;

bat_power_max = [redacted];
bat_power_min = -bat_power_max;

motor4Name = 'eRAD';
P4Data = load('Magna_eRAD.mat');
P4_eff = P4Data.eRAD_eff;
P4_rpm_axis = P4Data.eRAD_rpm_axis;
P4_torque_axis = P4Data.eRAD_torque_Axis;
EM4_max_torque = [redacted]; % [Nm] Parker 210 motor
EM4_max_power = [redacted];      % [Watts] max power of motor or
battery, whichever is limiting
EM4_gear_ratio = [redacted];    % Gear ratio between EM and wheels (no
final drive)
EM4_torque_min = -EM4_max_torque;
EM4_torque_max =  EM4_max_torque; %u2
EM4_power_max = EM4_max_power;
EM4_power_min = -EM4_max_power;

regen_efficiency_engine = 1;  % the efficiency of engine PPS
charging
regen_efficiency_braking = 1; % The efficiency of regenerative
braking, taken into account only braking at front can occur

battery_capacity = [redacted]; % energy capacity in kW-h
batt_res = [redacted]; % battery internal resistance [ohms]
batt_volt = 300; % battery voltage [V]

SOC_accuracy = 0.001;
SOC_low_range = 0.4;    % low desirable operating range for
battery
SOC_high_range = 0.7;   % high desirable operating range for
battery
```

```
SOC_final_low = 0.55-SOC_accuracy;     % low SOC at final step
(otherwise penalized)
SOC_final_high = 0.55+SOC_accuracy;    % high SOC at final step
(otherwise penalized)
SOC_start_low = 0.55-SOC_accuracy;    % low SOC at first step
(otherwise penalized)
SOC_start_high = 0.55+SOC_accuracy;   % high SOC at first step
(otherwise penalized)


SOC_min = SOC_low_range - SOC_accuracy; % These are the swept
parameters for SOC
SOC_max = SOC_high_range+SOC_accuracy;
engine_throttle_min = Engine_Fuel_Throttle_Axis(1);  %u1
engine_throttle_max = Engine_Fuel_Throttle_Axis(end);

SOC_penalty = 1000.0;    % proportional penalty parameter for
operating outside of SOC desirable (or final) range
NOT_ALLOWABLE_PENALTY = 1E+6;    % Used mostly as a prohibitive
cost when powertrain would be operated outside of an allowable
zone



%% Drive cycle



% Convert vehicle speed from mph to m/s and compute acceleration
v = Sch_Cycle(:,2)*0.44704;
t = Sch_Cycle(:,1);
a = diff(v)./diff(t);
totalDistance = trapz(v);

% Remove last data points for velocity and time since not present
in acceleration
v(end) = [];
t(end) = [];

[NUM_STEPS,~] = size(v);

% Calculate tractive force and power at each point in the drive
cycle
% force = fr*mass*g + 0.5*rho_air*Area*Cd* v.^2 + delta_M*mass*a;
force = A+B.*v+C.*v.^2+delta_M*mass*a;
power_required = force .* v+accessoryLoad;


%% Establish State and Control Grid Points

SOC_grid_size = state_GRID_size;
engine_throttle_grid_size = control_GRID_size;
alpha_grid_size = alpha_GRID_size;
```

```
GRID_SOC = linspace(SOC_min,SOC_max,SOC_grid_size)';
GRID_engine_throttle =
linspace(engine_throttle_min,engine_throttle_max,engine_throttle_
grid_size)';
GRID_alpha = linspace(0,1,alpha_grid_size);     %0 for 100% P1, 1
for 100% P4
% Evaluate the recursive cost starting at the last step in the
drive cycle
% and working backwards.  The only state variable is SOC, so the
cost
% function J_cost is size NUM_STEPS X SOC_grid_size.
% We also need to store the optimal control decisions at each
time step.
% This is given as U_store and is size NUM_STEPS X SOC_grid_size
X 3
% since we have three control variables (gear ratio index,
throttle request index,
% EM torque, EM4 torque, alpha index)

J_cost = zeros(NUM_STEPS,SOC_grid_size);
U_store = zeros(NUM_STEPS,SOC_grid_size,5);

% Array for storing the next SOC after a given step - used to
recover the
% optimal forward path. Stores the index that the (t = n) time
step/SOC
% wishes to go to at (t = n+1)
Next_SOC_index_store = zeros(NUM_STEPS,SOC_grid_size);

%
% First calculate J_cost(NUM_STEPS,:) - i.e., for all SOC in the
grid at
% the last point on the drive cycle.  Penalize any SOC not
between
% SOC_final_low and SOC_final_high. Note that the edited version
places a
% flat penalty on these regions
%
mask = GRID_SOC<SOC_final_low | GRID_SOC>SOC_final_high;
J_cost(end,mask) = SOC_penalty;


%
% Next do the cost at all other points on the drive cycle using
the
% recursive statement
%

next_step_SOC =
zeros(NUM_GEARS,engine_throttle_grid_size,alpha_grid_size);
```

```
engine_throttles =
GRID_engine_throttle'.*ones(NUM_GEARS,1,alpha_grid_size);
gearRatiosMatrix =
gear_ratios'.*ones(1,engine_throttle_grid_size,alpha_grid_size);
% vertical gear ratios, horizontal length (M)
zerosMat =
zeros(NUM_GEARS,engine_throttle_grid_size,alpha_grid_size);
onesMat =
ones(NUM_GEARS,engine_throttle_grid_size,alpha_grid_size);

for i = 1:alpha_grid_size
    alphas(1,1,i) = GRID_alpha(i);
end
alphas = alphas.*ones(NUM_GEARS,engine_throttle_grid_size);
GRID_SOC_2_1=GRID_SOC(2)-GRID_SOC(1);
for STEP = (NUM_STEPS-1):-1:1

    fprintf('Step %.0f of %.0f\n',STEP,NUM_STEPS);


[cost,battery_power_motor,battery_power_gen,EM_torque,EM4_torque]
= H_cost_CFree(v(STEP),power_required(STEP));
    mask1 = battery_power_motor>=0;
    mask2 = battery_power_gen<0 & engine_throttles==0;
    mask3 = battery_power_gen<0 & engine_throttles~=0;
    batter_power_m = battery_power_motor(mask1);
    batter_power_b = battery_power_gen(mask2);
    batter_power_pps = battery_power_gen(mask3);
    t_step_diff = t(STEP+1)-t(STEP);
    mask1_sub_factor =
batter_power_m.*(t_step_diff)./(battery_capacity.*1000.*3600);
    mask2_sub_factor =
regen_efficiency_braking.*batter_power_b.*(t_step_diff)./(battery
_capacity.*1000.*3600);
    mask3_sub_factor =
regen_efficiency_engine.*batter_power_pps.*(t_step_diff)./(batter
y_capacity.*1000.*3600);
    for L = 1:SOC_grid_size
        currentCost = cost;
        GRID_SOC_L=GRID_SOC(L);

        next_step_SOC(mask1) = GRID_SOC_L - mask1_sub_factor;

        next_step_SOC(mask2) = GRID_SOC_L - mask2_sub_factor;

        next_step_SOC(mask3) = GRID_SOC_L - mask3_sub_factor;

        mask = next_step_SOC < SOC_low_range | next_step_SOC >
SOC_high_range;
        currentCost(mask) = currentCost(mask)+SOC_penalty;
        next_step_SOC(next_step_SOC<SOC_low_range) =
SOC_low_range;
```

```
        next_step_SOC(next_step_SOC>SOC_high_range) =
SOC_high_range;


        SOC_dist = next_step_SOC-GRID_SOC_L;
        SOC_indexShift = round(SOC_dist./(GRID_SOC_2_1));
        next_SOC_index = L+SOC_indexShift;

        tempJ_cost = J_cost(STEP+1,:);  % creating this temporary
array is necessary for when we index J_cost in the following line
        nextCost = tempJ_cost(next_SOC_index);  %indexing like
this gives us the 3 dimensional matrix we need in nextCost

        currentCost = currentCost+nextCost;


        minCost = min(currentCost(:));
        [row,col,depth] =
ind2sub(size(currentCost),find(currentCost==minCost));
        r = row(1); c = col(1); d = depth(1);
            U_store(STEP,L,1) = r;
            U_store(STEP,L,2) = c;
            U_store(STEP,L,3) = EM_torque(r,c,d);
            U_store(STEP,L,4) = EM4_torque(r,c,d);
            U_store(STEP,L,5) = d;
            Next_SOC_index_store(STEP,L) = next_SOC_index(r,c,d);
        J_cost(STEP,L) = minCost;


    end
end


% Penalize any points that aren't within the starting bounds
mask = GRID_SOC<SOC_start_low | GRID_SOC>SOC_start_high;
J_cost(1,mask) = J_cost(1,mask)+SOC_penalty;

%% Post process

%
% Next, move through the cycle and recover the optimal solution
% working forwards from min(J_cost(1,:))
%
FORWARD_cost = zeros(NUM_STEPS,1);
FORWARD_gear = zeros(NUM_STEPS,1);
FORWARD_throttle_request = zeros(NUM_STEPS,1);
FORWARD_EM_torque = zeros(NUM_STEPS,1);
FORWARD_EM4_torque = zeros(NUM_STEPS,1);
FORWARD_cost_index = zeros(NUM_STEPS,1);
FORWARD_SOC = zeros(NUM_STEPS,1);
FORWARD_alpha = zeros(NUM_STEPS,1);
```

```
% First step
[FORWARD_cost(1),FORWARD_cost_index(1)] = min(J_cost(1,:)); %
find the minimum cost for the first step and the corresponding
SOC index
FORWARD_SOC(1) = GRID_SOC(FORWARD_cost_index(1)); % the SOC where
the cost at step 1 is minimal

% Steps 2 through NUM_STEPS-1
for STEP = 2:NUM_STEPS-1
    FORWARD_cost_index(STEP) = Next_SOC_index_store(STEP-
1,FORWARD_cost_index(STEP-1)); % STEP-1 stores the next SOC index
to go to in the variable Next_SOC_index_store
    FORWARD_cost(STEP) = J_cost(STEP, FORWARD_cost_index(STEP));
    FORWARD_SOC(STEP) = GRID_SOC(FORWARD_cost_index(STEP));
    FORWARD_gear(STEP) = U_store(STEP,
FORWARD_cost_index(STEP),1);
    FORWARD_throttle_request(STEP) =
GRID_engine_throttle(U_store(STEP, FORWARD_cost_index(STEP),2));
    FORWARD_EM_torque(STEP) = U_store(STEP,
FORWARD_cost_index(STEP),3);
    FORWARD_EM4_torque(STEP) = U_store(STEP,
FORWARD_cost_index(STEP),4);
    FORWARD_alpha(STEP) = GRID_alpha(U_store(STEP,
FORWARD_cost_index(STEP),5));
end

% Last step
FORWARD_cost_index(NUM_STEPS) = Next_SOC_index_store(NUM_STEPS-
1,FORWARD_cost_index(NUM_STEPS-1));
FORWARD_cost(NUM_STEPS) = J_cost(NUM_STEPS,
FORWARD_cost_index(NUM_STEPS));
FORWARD_SOC(NUM_STEPS) = GRID_SOC(FORWARD_cost_index(NUM_STEPS));


driveRange = 1:NUM_STEPS;

figure;
subplot(3,1,1)
yyaxis left
plot(driveRange,v,driveRange,FORWARD_gear)
yyaxis right
plot(driveRange,FORWARD_alpha.*100,driveRange,100.*FORWARD_throtte_request./Engine_Torque_Throttle_Axis(end));
title('Drive Cycle Dynamics')
xlabel('Drive Cycle Index/Time [s]')
legend('Velocity (m/s)','Gear','alpha','ICE Throttle %')

subplot(3,1,2)
plot(driveRange,100.*FORWARD_EM_torque./EM_torque_max,driveRange,
100.*FORWARD_EM4_torque./EM4_torque_max)
legend('EM Throttle %','EM4 Throttle %')
xlabel('Drive Cycle Index/Time [s]')
```

```
subplot(3,1,3)
plot(1:NUM_STEPS,FORWARD_SOC);
title('State of Charge over the Drive Cycle')
ylabel('State of Charge')
xlabel('Drive Cycle Index/Time [s]')


%% MPG
FORWARD_gear(FORWARD_gear==0) = 1; % gear starts and ends at
zero, which gear_ratios doesn't like
CycleICERPM = v.*i_final.*gear_ratios(FORWARD_gear)'./radius .*
60./(2.*pi);
if sum(CycleICERPM > max_engine_RPM) > 0
    CycleICERPM(CycleICERPM > max_engine_RPM) = max_engine_RPM; %
this is a problem when interpolating
    warning('Some ICE points had higher than allowed RPMs')
end
CycleFuelUsage =
interp2(Engine_Fuel_RPM_Axis,Engine_Fuel_Throttle_Axis,Engine_Fue
l_Data,CycleICERPM, FORWARD_throttle_request); %grams per second
fuel use
TotalFuelUsed=sum(sum(CycleFuelUsage)); %grams
FuelVolumeUsed = TotalFuelUsed/1000 / FuelDensity;    %Liters
FuelVUsedGal = FuelVolumeUsed*.264172;  %[gallons] fuel
Lper100km = FuelVolumeUsed./(totalDistance/1000) * 100;
Dmiles = totalDistance/1000 *.621371; % total distance in miles
mpg = Dmiles/FuelVUsedGal;
fprintf('Miles per Gallon, petrol only: %.1f \n',mpg)

timeElapsed = toc

save(['Results\P1P4_' cycleFileName '_' engineFileName '_'
motorName '_' motor4Name '_' num2str(state_GRID_size) 'x'
num2str(control_GRID_size) 'x' num2str(alpha_GRID_size)
'pts.mat'])



%% cost function
function
[H_cost,battery_power_motor,battery_power_gen,EM_torque,EM4_torqu
e] = H_cost_CFree(vel,power_required)
% Cost function for the Dynamic Programming HEV script
% Written by Michael J. Leamy, August, 2011
% Edited by Christian Free, August 2019
H_cost = zerosMat;
rpms = (vel*i_final/radius * 60/(2*pi)).*gear_ratios';
rpms = rpms.*onesMat;
```

```
EM_rpm =
vel./(2.*pi.*radius).*gearRatiosMatrix.*EM_gear_ratio.*i_final
.*60;
EM4_rpm = vel./(2.*pi.*radius).*EM4_gear_ratio.*60 .*onesMat;

engine_torque =
interp2(Engine_Torque_RPM_Axis,Engine_Torque_Throttle_Axis,
Engine_Torque_Data,rpms,engine_throttles);
% transmission efficiency:

transrpms = rpms; transrpms(rpms>max(omega_bpts)) =
max(omega_bpts); transrpms(rpms<min(omega_bpts)) =
min(omega_bpts);
transTorq = engine_torque; transTorq(transTorq>max(Trq_bpts)) =
max(Trq_bpts); transTorq(transTorq<min(Trq_bpts)) =
min(Trq_bpts);
eta =
interp3(omega_bpts,Trq_bpts,gear_ratios,eta_tbl,transrpms,transTo
rq,gear_ratios'.*onesMat);
engine_power = engine_torque .* rpms .* 2.*pi./60 .* eta; %
engine power in Watts, (eta=transmission efficiency)
battery_request = power_required-engine_power;      % battery
power requested


% here we clip battery/motor power based on hardware limits:
battery_request(battery_request>bat_power_max) = bat_power_max;
battery_request(battery_request<bat_power_min) = bat_power_min;

EM_power = battery_request.*(1-alphas);
EM4_power = battery_request.*(alphas);

EM_power(EM_power>EM_power_max) = EM_power_max;
EM_power(EM_power<EM_power_min) = EM_power_min;
EM4_power(EM4_power>EM4_power_max) = EM4_power_max;
EM4_power(EM4_power<EM4_power_min) = EM4_power_min;


if vel == 0
    EM_torque = zerosMat;
    EM4_torque = zerosMat;
else
    EM_torque = EM_power./(2.*pi./60.*EM_rpm);
    EM4_torque = EM4_power./(2.*pi./60.*EM4_rpm);
end



% here we clip motor torque based on hardware limits:
% If the motor isn't providing enough tractive effort, it will
later see
% a (battery_power + engine_power) < power_required penalty
```

```
%If the motor is being asked to regen more than it can, it
recieves no
%penalty, but has its torque clipped. During PPS charging w/
engine, there will be a
%natural fuel penalty to regening more than the motor can take
EM_torque(EM_torque>EM_torque_max) = EM_torque_max;
EM_torque(EM_torque<EM_torque_min) = EM_torque_min;
EM4_torque(EM4_torque>EM4_torque_max) = EM4_torque_max;
EM4_torque(EM4_torque<EM4_torque_min) = EM4_torque_min;

EM_power = EM_torque .* (2.*pi./60.*EM_rpm);
EM4_power = EM4_torque .* (2.*pi./60.*EM4_rpm);

EM_eff =
interp2(Denso_rpm_axis,Denso_torque_axis,Denso_eff,EM_rpm,abs(EM_
torque));
EM4_eff =
interp2(P4_rpm_axis,P4_torque_axis,P4_eff,EM4_rpm,abs(EM4_torque)
);
battery_power_raw = EM_power+EM4_power;
battery_power_gen = EM_power.*EM_eff+EM4_power.*EM4_eff;
battery_power_motor = EM_power./EM_eff +EM4_power./EM4_eff;

% add special case where accessory load is only load
if vel ==0
    battery_power_raw = accessoryLoad.*1000.*onesMat;
    battery_power_gen = battery_power_raw;
    battery_power_motor = battery_power_raw;
end

% add power due to battery internal resistance
batt_current_motor = battery_power_raw./batt_volt;
battery_power_motor = battery_power_motor+(batt_current_motor.^2
.*batt_res);

batt_current_gen = battery_power_gen./batt_volt;
battery_power_gen = battery_power_gen+(batt_current_gen.^2
.*batt_res);


%
% Compute cost using the grams/sec from the fuel table
% Make inoperable regions very costly
%
if(power_required <= 0)

    % When power_required <= 0 we penalize heavily any
    % usage of the ICE so that we only operate in
    % regenerative braking, hybrid braking mode, or
    % mechanical braking mode.  Note that at present this doesn't
allow
    % battery charging mode from engine while braking.
```

```
    H_cost = H_cost+battery_power_gen*1e-9;     % reward regen
lightly to favor the configuration that generates more power
    H_cost(engine_throttles>0) = NOT_ALLOWABLE_PENALTY;
    %H_cost(GRID_engine_throttle== 0) is already = 0, or
penalized
else

    % Penalize any inoperable conditions
    %    Engine RPM greater than allowable
    %    Battery plus engine power not meeting power required
    %    Engine RPM less than 750 and a gear higher than first
gear requested (reason being no non-zero fuel data below 750)
    %
    % Note: for low speeds v and large power
    % required, we can fail to produce enough power
    % from EM to overcome negative power of ICE
    % while meeting the required power.  In this
    % case, the EM torque needs to be increased, or
    % logic must be added to allow ICE to rev and
    % exceed the rpm value (think of as a
    % slipping clutch)
    %
    % fix for the event that EM and ICE can't meet goal
    H_cost = interp2(Engine_Fuel_RPM_Axis,
Engine_Fuel_Throttle_Axis, Engine_Fuel_Data, rpms,
engine_torque);
    H_cost = H_cost+battery_power_motor*1e-9; %penalize using
unnecessary power, such as keeping the engine in a higher gear at
0 throttle.
    H_cost(rpms<750) = interp2(Engine_Fuel_RPM_Axis,
Engine_Fuel_Throttle_Axis, Engine_Fuel_Data, 750,
engine_torque(rpms<750));
    mask = rpms<750 & gearRatiosMatrix<gearRatiosMatrix(1,1);
    H_cost(mask) = H_cost(mask)+NOT_ALLOWABLE_PENALTY;
    mask = (battery_power_raw + engine_power +1e-9) <
power_required;    %1e-9 to leave room for computational error
    H_cost(mask) = H_cost(mask)+NOT_ALLOWABLE_PENALTY +
0.1.*(power_required-battery_power_raw(mask)-engine_power(mask));
    mask = rpms>4000;
    H_cost(mask) = H_cost(mask)+40;
end

mask = rpms>max_engine_RPM;
H_cost(mask) = H_cost(mask) + NOT_ALLOWABLE_PENALTY;
battery_power_gen(mask) = inf;
battery_power_motor(mask) = inf;
EM_torque(mask) = inf;
end

end
```

REFERENCES

[1] Ehsani, M. and Gao, Y. and Emadi, A., 2010, *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles*, 2nd ed., CRC Press, Boca Raton, FL, Fig. 2.30

[2] S. Onori, L. Serrao and G. Rizzoni, "Hybrid Electric Vehicles: Energy Management Strategies," Springer Briefs in Electrical and Computer Engineering, pp. 65-87, 2016

[3] Schaich, N., 2020, "Characterization of a Hybrid Electric Mobility as a Service Vehicle," M.S. Thesis, Mechanical Engineering, Georgia Institute of Technology

[4] Pei, D. and Leamy, M., 2013, "Dynamic Programming-Informed Equivalent Cost Minimization Control Strategies for Hybrid-Electric Vehicles," Journal of Dynamic Systems, Measurement, and Control, **vol. 135**

[5] Mangino Buick GMC, 11 May 2020, "Suspension Subframe Crossmember (Rear) 84660445," gm-wholseale-parts.com, from https://www.gm-wholesale-parts.com/p/Chevrolet__Blazer/Suspension-Subframe-Crossmember-Rear/91021052/84660445.html?partner=googlebase_adwords&kwd=&origin=pla&partnerDevice=c&userLocation=9010937&gclid=CjwKCAjwwMn1BRAUEiwAZ_jnEvCcCMTn3kQ3V_1BK3fxo96eaD-eQ-tvOliasqOTQ8umgNW3ohez7xoC5FsQAvD_BwE

[6] Budynas, R. and Nisbett, J., 2015, *Shigley's Mechanical Engineering Design*, 10th ed., McGraw-Hill Education, New York, NY, Chap. 6